Data Structures and Algorithms in Java

Procedural Programming: Control Flow

Outline

If Statements

If Statements

```
if (<boolean-expression>) {
    <statement>
    ...
} else if (<boolean-expression>) {
    <statement>
    ...
} else if (<boolean-expression>) {
    <statement>
    ...
...
} else {
    <statement>
    ...
}
```

If Statements · Example (Letter Grade)

If Statements · Example (Letter Grade)

Grade.java

Command-line input	a percentage *score* (double)
Standard output	the corresponding letter grade

If Statements · Example (Letter Grade)

Grade.java

Command-line input	a percentage *score* (double)
Standard output	the corresponding letter grade

~/workspace/dsaj

```
$ _
```

If Statements · Example (Letter Grade)

Grade.java

Command-line input	a percentage *score* (double)
Standard output	the corresponding letter grade

~/workspace/dsaj

```
$ java Grade 97
```

If Statements · Example (Letter Grade)

Grade.java

Command-line input	a percentage *score* (double)
Standard output	the corresponding letter grade

~/workspace/dsaj

```
$ java Grade 97
A
$ _
```

If Statements · Example (Letter Grade)

Grade.java

Command-line input	a percentage *score* (double)
Standard output	the corresponding letter grade

~/workspace/dsaj

```
$ java Grade 97
A
$ java Grade 56
```

If Statements · Example (Letter Grade)

Grade.java

Command-line input	a percentage *score* (double)
Standard output	the corresponding letter grade

~/workspace/dsaj

```
$ java Grade 97
A
$ java Grade 56
F
$ _
```

If Statements · Example (Letter Grade)

If Statements · Example (Letter Grade)

Grade.java

```java
import stdlib.StdOut;

public class Grade {
    public static void main(String[] args) {
        double score = Double.parseDouble(args[0]);
        if (score >= 93) {
            StdOut.println("A");
        } else if (score >= 90) {
            StdOut.println("A-");
        } else if (score >= 87) {
            StdOut.println("B+");
        } else if (score >= 83) {
            StdOut.println("B");
        } else if (score >= 80) {
            StdOut.println("B-");
        } else if (score >= 77) {
            StdOut.println("C+");
        } else if (score >= 73) {
            StdOut.println("C");
        } else if (score >= 70) {
            StdOut.println("C-");
        } else if (score >= 67) {
            StdOut.println("D+");
        } else if (score >= 63) {
            StdOut.println("D");
        } else if (score >= 60) {
            StdOut.println("D-");
        } else {
            StdOut.println("F");
        }
    }
}
```

If Statements · Example (Letter Grade)

If Statements · Example (Letter Grade)

```java
import stdlib.StdOut;

public class Grade {
    public static void main(String[] args) {
        double score = Double.parseDouble(args[0]);
        if (score >= 93) {
            StdOut.println("A");
        } else if (score >= 90) {
            StdOut.println("A-");
        } else if (score >= 87) {
            StdOut.println("B+");
        } else if (score >= 83) {
            StdOut.println("B");
        } else if (score >= 80) {
            StdOut.println("B-");
        } else if (score >= 77) {
            StdOut.println("C+");
        } else if (score >= 73) {
            StdOut.println("C");
        } else if (score >= 70) {
            StdOut.println("C-");
        } else if (score >= 67) {
            StdOut.println("D+");
        } else if (score >= 63) {
            StdOut.println("D");
        } else if (score >= 60) {
            StdOut.println("D-");
        } else {
            StdOut.println("F");
        }
    }
}
```

Variable Trace

line #	score

>_

```
$ _
```

If Statements · Example (Letter Grade)

```java
import stdlib.StdOut;

public class Grade {
    public static void main(String[] args) {
        double score = Double.parseDouble(args[0]);
        if (score >= 93) {
            StdOut.println("A");
        } else if (score >= 90) {
            StdOut.println("A-");
        } else if (score >= 87) {
            StdOut.println("B+");
        } else if (score >= 83) {
            StdOut.println("B");
        } else if (score >= 80) {
            StdOut.println("B-");
        } else if (score >= 77) {
            StdOut.println("C+");
        } else if (score >= 73) {
            StdOut.println("C");
        } else if (score >= 70) {
            StdOut.println("C-");
        } else if (score >= 67) {
            StdOut.println("D+");
        } else if (score >= 63) {
            StdOut.println("D");
        } else if (score >= 60) {
            StdOut.println("D-");
        } else {
            StdOut.println("F");
        }
    }
}
```

Variable Trace

line #	score

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1  import stdlib.StdOut;
 2
 3  public class Grade {
 4      public static void main(String[] args) {
 5          double score = Double.parseDouble(args[0]);
 6          if (score >= 93) {
 7              StdOut.println("A");
 8          } else if (score >= 90) {
 9              StdOut.println("A-");
10          } else if (score >= 87) {
11              StdOut.println("B+");
12          } else if (score >= 83) {
13              StdOut.println("B");
14          } else if (score >= 80) {
15              StdOut.println("B-");
16          } else if (score >= 77) {
17              StdOut.println("C+");
18          } else if (score >= 73) {
19              StdOut.println("C");
20          } else if (score >= 70) {
21              StdOut.println("C-");
22          } else if (score >= 67) {
23              StdOut.println("D+");
24          } else if (score >= 63) {
25              StdOut.println("D");
26          } else if (score >= 60) {
27              StdOut.println("D-");
28          } else {
29              StdOut.println("F");
30          }
31      }
32  }
```

Variable Trace

line #	score
1	

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1 import stdlib.StdOut;
 2
 3 public class Grade {
 4     public static void main(String[] args) {
 5         double score = Double.parseDouble(args[0]);
 6         if (score >= 93) {
 7             StdOut.println("A");
 8         } else if (score >= 90) {
 9             StdOut.println("A-");
10         } else if (score >= 87) {
11             StdOut.println("B+");
12         } else if (score >= 83) {
13             StdOut.println("B");
14         } else if (score >= 80) {
15             StdOut.println("B-");
16         } else if (score >= 77) {
17             StdOut.println("C+");
18         } else if (score >= 73) {
19             StdOut.println("C");
20         } else if (score >= 70) {
21             StdOut.println("C-");
22         } else if (score >= 67) {
23             StdOut.println("D+");
24         } else if (score >= 63) {
25             StdOut.println("D");
26         } else if (score >= 60) {
27             StdOut.println("D-");
28         } else {
29             StdOut.println("F");
30         }
31     }
32 }
```

Variable Trace

line #	score
3	

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1 import stdlib.StdOut;
 2 
 3 public class Grade {
 4     public static void main(String[] args) {
 5         double score = Double.parseDouble(args[0]);
 6         if (score >= 93) {
 7             StdOut.println("A");
 8         } else if (score >= 90) {
 9             StdOut.println("A-");
10         } else if (score >= 87) {
11             StdOut.println("B+");
12         } else if (score >= 83) {
13             StdOut.println("B");
14         } else if (score >= 80) {
15             StdOut.println("B-");
16         } else if (score >= 77) {
17             StdOut.println("C+");
18         } else if (score >= 73) {
19             StdOut.println("C");
20         } else if (score >= 70) {
21             StdOut.println("C-");
22         } else if (score >= 67) {
23             StdOut.println("D+");
24         } else if (score >= 63) {
25             StdOut.println("D");
26         } else if (score >= 60) {
27             StdOut.println("D-");
28         } else {
29             StdOut.println("F");
30         }
31     }
32 }
```

Variable Trace

line #	score
4	

```
>_
$ java Grade 82
```

If Statements · Example (Letter Grade)

```java
import stdlib.StdOut;

public class Grade {
    public static void main(String[] args) {
        double score = Double.parseDouble(args[0]);
        if (score >= 93) {
            StdOut.println("A");
        } else if (score >= 90) {
            StdOut.println("A-");
        } else if (score >= 87) {
            StdOut.println("B+");
        } else if (score >= 83) {
            StdOut.println("B");
        } else if (score >= 80) {
            StdOut.println("B-");
        } else if (score >= 77) {
            StdOut.println("C+");
        } else if (score >= 73) {
            StdOut.println("C");
        } else if (score >= 70) {
            StdOut.println("C-");
        } else if (score >= 67) {
            StdOut.println("D+");
        } else if (score >= 63) {
            StdOut.println("D");
        } else if (score >= 60) {
            StdOut.println("D-");
        } else {
            StdOut.println("F");
        }
    }
}
```

Variable Trace

line #	score
5	82.0

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1 import stdlib.StdOut;
 2
 3 public class Grade {
 4     public static void main(String[] args) {
 5         double score = Double.parseDouble(args[0]);
 6         if (score >= 93) {
 7             StdOut.println("A");
 8         } else if (score >= 90) {
 9             StdOut.println("A-");
10         } else if (score >= 87) {
11             StdOut.println("B+");
12         } else if (score >= 83) {
13             StdOut.println("B");
14         } else if (score >= 80) {
15             StdOut.println("B-");
16         } else if (score >= 77) {
17             StdOut.println("C+");
18         } else if (score >= 73) {
19             StdOut.println("C");
20         } else if (score >= 70) {
21             StdOut.println("C-");
22         } else if (score >= 67) {
23             StdOut.println("D+");
24         } else if (score >= 63) {
25             StdOut.println("D");
26         } else if (score >= 60) {
27             StdOut.println("D-");
28         } else {
29             StdOut.println("F");
30         }
31     }
32 }
```

Variable Trace

line #	score
6	82.0

```
>_
$ java Grade 82
```

If Statements · Example (Letter Grade)

```java
import stdlib.StdOut;

public class Grade {
    public static void main(String[] args) {
        double score = Double.parseDouble(args[0]);
        if (score >= 93) {
            StdOut.println("A");
        } else if (score >= 90) {
            StdOut.println("A-");
        } else if (score >= 87) {
            StdOut.println("B+");
        } else if (score >= 83) {
            StdOut.println("B");
        } else if (score >= 80) {
            StdOut.println("B-");
        } else if (score >= 77) {
            StdOut.println("C+");
        } else if (score >= 73) {
            StdOut.println("C");
        } else if (score >= 70) {
            StdOut.println("C-");
        } else if (score >= 67) {
            StdOut.println("D+");
        } else if (score >= 63) {
            StdOut.println("D");
        } else if (score >= 60) {
            StdOut.println("D-");
        } else {
            StdOut.println("F");
        }
    }
}
```

Variable Trace

line #	score
8	82.0

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1  import stdlib.StdOut;
 2
 3  public class Grade {
 4      public static void main(String[] args) {
 5          double score = Double.parseDouble(args[0]);
 6          if (score >= 93) {
 7              StdOut.println("A");
 8          } else if (score >= 90) {
 9              StdOut.println("A-");
10          } else if (score >= 87) {
11              StdOut.println("B+");
12          } else if (score >= 83) {
13              StdOut.println("B");
14          } else if (score >= 80) {
15              StdOut.println("B-");
16          } else if (score >= 77) {
17              StdOut.println("C+");
18          } else if (score >= 73) {
19              StdOut.println("C");
20          } else if (score >= 70) {
21              StdOut.println("C-");
22          } else if (score >= 67) {
23              StdOut.println("D+");
24          } else if (score >= 63) {
25              StdOut.println("D");
26          } else if (score >= 60) {
27              StdOut.println("D-");
28          } else {
29              StdOut.println("F");
30          }
31      }
32  }
```

Variable Trace

line #	score
10	82.0

```
>_
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1 import stdlib.StdOut;
 2
 3 public class Grade {
 4     public static void main(String[] args) {
 5         double score = Double.parseDouble(args[0]);
 6         if (score >= 93) {
 7             StdOut.println("A");
 8         } else if (score >= 90) {
 9             StdOut.println("A-");
10         } else if (score >= 87) {
11             StdOut.println("B+");
12         } else if (score >= 83) {
13             StdOut.println("B");
14         } else if (score >= 80) {
15             StdOut.println("B-");
16         } else if (score >= 77) {
17             StdOut.println("C+");
18         } else if (score >= 73) {
19             StdOut.println("C");
20         } else if (score >= 70) {
21             StdOut.println("C-");
22         } else if (score >= 67) {
23             StdOut.println("D+");
24         } else if (score >= 63) {
25             StdOut.println("D");
26         } else if (score >= 60) {
27             StdOut.println("D-");
28         } else {
29             StdOut.println("F");
30         }
31     }
32 }
```

Variable Trace

line #	score
12	82.0

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1 import stdlib.StdOut;
 2 
 3 public class Grade {
 4     public static void main(String[] args) {
 5         double score = Double.parseDouble(args[0]);
 6         if (score >= 93) {
 7             StdOut.println("A");
 8         } else if (score >= 90) {
 9             StdOut.println("A-");
10         } else if (score >= 87) {
11             StdOut.println("B+");
12         } else if (score >= 83) {
13             StdOut.println("B");
14         } else if (score >= 80) {
15             StdOut.println("B-");
16         } else if (score >= 77) {
17             StdOut.println("C+");
18         } else if (score >= 73) {
19             StdOut.println("C");
20         } else if (score >= 70) {
21             StdOut.println("C-");
22         } else if (score >= 67) {
23             StdOut.println("D+");
24         } else if (score >= 63) {
25             StdOut.println("D");
26         } else if (score >= 60) {
27             StdOut.println("D-");
28         } else {
29             StdOut.println("F");
30         }
31     }
32 }
```

Variable Trace

line #	score
14	82.0

```
$ java Grade 82
```

If Statements · Example (Letter Grade)

```
 1  import stdlib.StdOut;
 2
 3  public class Grade {
 4      public static void main(String[] args) {
 5          double score = Double.parseDouble(args[0]);
 6          if (score >= 93) {
 7              StdOut.println("A");
 8          } else if (score >= 90) {
 9              StdOut.println("A-");
10          } else if (score >= 87) {
11              StdOut.println("B+");
12          } else if (score >= 83) {
13              StdOut.println("B");
14          } else if (score >= 80) {
15              StdOut.println("B-");
16          } else if (score >= 77) {
17              StdOut.println("C+");
18          } else if (score >= 73) {
19              StdOut.println("C");
20          } else if (score >= 70) {
21              StdOut.println("C-");
22          } else if (score >= 67) {
23              StdOut.println("D+");
24          } else if (score >= 63) {
25              StdOut.println("D");
26          } else if (score >= 60) {
27              StdOut.println("D-");
28          } else {
29              StdOut.println("F");
30          }
31      }
32  }
```

Variable Trace

line #	score
15	82.0

>_

```
$ java Grade 82
B-
```

If Statements · Example (Letter Grade)

```java
 1 import stdlib.StdOut;
 2 
 3 public class Grade {
 4     public static void main(String[] args) {
 5         double score = Double.parseDouble(args[0]);
 6         if (score >= 93) {
 7             StdOut.println("A");
 8         } else if (score >= 90) {
 9             StdOut.println("A-");
10         } else if (score >= 87) {
11             StdOut.println("B+");
12         } else if (score >= 83) {
13             StdOut.println("B");
14         } else if (score >= 80) {
15             StdOut.println("B-");
16         } else if (score >= 77) {
17             StdOut.println("C+");
18         } else if (score >= 73) {
19             StdOut.println("C");
20         } else if (score >= 70) {
21             StdOut.println("C-");
22         } else if (score >= 67) {
23             StdOut.println("D+");
24         } else if (score >= 63) {
25             StdOut.println("D");
26         } else if (score >= 60) {
27             StdOut.println("D-");
28         } else {
29             StdOut.println("F");
30         }
31     }
32 }
```

Variable Trace

line #	score
15	82.0

```
>_
$ java Grade 82
B-
$ _
```

If Statements · Example (Letter Grade)

```java
import stdlib.StdOut;

public class Grade {
    public static void main(String[] args) {
        double score = Double.parseDouble(args[0]);
        if (score >= 93) {
            StdOut.println("A");
        } else if (score >= 90) {
            StdOut.println("A-");
        } else if (score >= 87) {
            StdOut.println("B+");
        } else if (score >= 83) {
            StdOut.println("B");
        } else if (score >= 80) {
            StdOut.println("B-");
        } else if (score >= 77) {
            StdOut.println("C+");
        } else if (score >= 73) {
            StdOut.println("C");
        } else if (score >= 70) {
            StdOut.println("C-");
        } else if (score >= 67) {
            StdOut.println("D+");
        } else if (score >= 63) {
            StdOut.println("D");
        } else if (score >= 60) {
            StdOut.println("D-");
        } else {
            StdOut.println("F");
        }
    }
}
```

Variable Trace

line #	score

```
>_
$ java Grade 82
B-
$ _
```

Conditional Expressions

Conditional Expressions

```
<boolean-expression> ? <then-expression> : <else-expression>
```

Conditional Expressions · Example (Coin Flip)

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output	“Heads” or “Tails”

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output | “Heads” or “Tails”

~/workspace/dsaj

```
$ _
```

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output	“Heads” or “Tails”

~/workspace/dsaj

```
$ java Flip
```

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output	“Heads” or “Tails”

~/workspace/dsaj

```
$ java Flip
Heads
$ _
```

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output | “Heads” or “Tails”

~/workspace/dsaj

```
$ java Flip
Heads
$ java Flip
```

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output	"Heads" or "Tails"

~/workspace/dsaj

```
$ java Flip
Heads
$ java Flip
Heads
$ _
```

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output	“Heads” or “Tails”

~/workspace/dsaj

```
$ java Flip
Heads
$ java Flip
Heads
$ java Flip
```

Conditional Expressions · Example (Coin Flip)

Flip.java

Standard output	“Heads” or “Tails”

~/workspace/dsaj

```
$ java Flip
Heads
$ java Flip
Heads
$ java Flip
Tails
$ _
```

Conditional Expressions · Example (Coin Flip)

Conditional Expressions · Example (Coin Flip)

Flip.java

```java
import stdlib.StdOut;
import stdlib.StdRandom;

public class Flip {
    public static void main(String[] args) {
        String result = StdRandom.bernoulli() ? "Heads" : "Tails";
        StdOut.println(result);
    }
}
```

While Statements

While Statements

```
while (<boolean-expression>) {
    <statement>
    ...
}
```

While Statements · Example (N Hellos)

While Statements · Example (*N* Hellos)

`NHellos.java`

Command-line input	*n* (int)
Standard output	*n* hellos

While Statements · Example (N Hellos)

NHellos.java

Command-line input	n (int)
Standard output	n hellos

~/workspace/dsaj

```
$ _
```

While Statements · Example (*N* Hellos)

NHellos.java

Command-line input	*n* (int)
Standard output	*n* hellos

~/workspace/dsaj

```
$ java NHellos 10
```

While Statements · Example (*N* Hellos)

NHellos.java

Command-line input	*n* (int)
Standard output	*n* hellos

~/workspace/dsaj

```
$ java NHellos 10
Hello # 1
Hello # 2
Hello # 3
Hello # 4
Hello # 5
Hello # 6
Hello # 7
Hello # 8
Hello # 9
Hello # 10
$ _
```

While Statements · Example (*N* Hellos)

While Statements · Example (*N* Hellos)

NHellos.java

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

While Statements · Example (N Hellos)

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i

```
>_
$ _
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i

```
>_
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```
 1 import stdlib.StdOut;
 2 
 3 public class NHellos {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         int i = 1;
 7         while (i <= n) {
 8             StdOut.println("Hello # " + i);
 9             i++;
10         }
11     }
12 }
```

Variable Trace

line #	n	i
1		

```
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```
 1  import stdlib.StdOut;
 2
 3  public class NHellos {
 4      public static void main(String[] args) {
 5          int n = Integer.parseInt(args[0]);
 6          int i = 1;
 7          while (i <= n) {
 8              StdOut.println("Hello # " + i);
 9              i++;
10          }
11      }
12  }
```

Variable Trace

line #	n	i
3		

```
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
4		

```
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
5	3	

```
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```
 1 import stdlib.StdOut;
 2
 3 public class NHellos {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         int i = 1;
 7         while (i <= n) {
 8             StdOut.println("Hello # " + i);
 9             i++;
10         }
11     }
12 }
```

Variable Trace

line #	n	i
6	3	1

```
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
7	3	1

```
$ java NHellos 3
```

While Statements · Example (*N* Hellos)

```
 1 import stdlib.StdOut;
 2 
 3 public class NHellos {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         int i = 1;
 7         while (i <= n) {
 8             StdOut.println("Hello # " + i);
 9             i++;
10         }
11     }
12 }
```

Variable Trace

line #	n	i
8	3	1

```
$ java NHellos 3
Hello # 1
```

While Statements · Example (*N* Hellos)

```
 1 import stdlib.StdOut;
 2
 3 public class NHellos {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         int i = 1;
 7         while (i <= n) {
 8             StdOut.println("Hello # " + i);
 9             i++;
10         }
11     }
12 }
```

Variable Trace

line #	n	i
9	3	2

```
$ java NHellos 3
Hello # 1
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
7	3	2

```
$ java NHellos 3
Hello # 1
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
8	3	2

```
$ java NHellos 3
Hello # 1
Hello # 2
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
9	3	3

```
$ java NHellos 3
Hello # 1
Hello # 2
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
7	3	3

```
$ java NHellos 3
Hello # 1
Hello # 2
```

While Statements · Example (*N* Hellos)

```
 1 import stdlib.StdOut;
 2
 3 public class NHellos {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         int i = 1;
 7         while (i <= n) {
 8             StdOut.println("Hello # " + i);
 9             i++;
10         }
11     }
12 }
```

Variable Trace

line #	n	i
8	3	3

```
$ java NHellos 3
Hello # 1
Hello # 2
Hello # 3
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i
8	3	4

```
$ java NHellos 3
Hello # 1
Hello # 2
Hello # 3
```

While Statements · Example (*N* Hellos)

```
 1 import stdlib.StdOut;
 2
 3 public class NHellos {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         int i = 1;
 7         while (i <= n) {
 8             StdOut.println("Hello # " + i);
 9             i++;
10         }
11     }
12 }
```

Variable Trace

line #	n	i
7	3	4

>_

```
$ java NHellos 3
Hello # 1
Hello # 2
Hello # 3
```

While Statements · Example (*N* Hellos)

```java
import stdlib.StdOut;

public class NHellos {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int i = 1;
        while (i <= n) {
            StdOut.println("Hello # " + i);
            i++;
        }
    }
}
```

Variable Trace

line #	n	i

```
>_
$ java NHellos 3
Hello # 1
Hello # 2
Hello # 3
$ _
```

For Statements

For Statements

```
for (<initialization>; <boolean-expression>; <update>) {
    <statement>
    ...
}
```

For Statements · Example (Harmonic Numbers)

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ _
```

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ java Harmonic 10
```

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ java Harmonic 10
2.9289682539682538
$ _
```

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ java Harmonic 10
2.9289682539682538
$ java Harmonic 1000
```

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ java Harmonic 10
2.9289682539682538
$ java Harmonic 1000
7.485470860550343
$ _
```

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ java Harmonic 10
2.9289682539682538
$ java Harmonic 1000
7.485470860550343
$ java Harmonic 10000
```

For Statements · Example (Harmonic Numbers)

Harmonic.java

Command-line input	n (int)
Standard output	the nth harmonic number, $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \ln(n) + 0.57721$

~/workspace/dsaj

```
$ java Harmonic 10
2.9289682539682538
$ java Harmonic 1000
7.485470860550343
$ java Harmonic 10000
9.787606036044348
$ _
```

For Statements · Example (Harmonic Numbers)

For Statements · Example (Harmonic Numbers)

Harmonic.java

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

For Statements · Example (Harmonic Numbers)

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i

```
$ _
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```
 1 import stdlib.StdOut;
 2 
 3 public class Harmonic {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         double total = 0.0;
 7         for (int i = 1; i <= n; i++) {
 8             total += 1.0 / i;
 9         }
10         StdOut.println(total);
11     }
12 }
```

Variable Trace

line #	n	total	i
1			

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```
 1 import stdlib.StdOut;
 2
 3 public class Harmonic {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         double total = 0.0;
 7         for (int i = 1; i <= n; i++) {
 8             total += 1.0 / i;
 9         }
10         StdOut.println(total);
11     }
12 }
```

Variable Trace

line #	n	total	i
3			

```
>_
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
1  import stdlib.StdOut;
2
3  public class Harmonic {
4      public static void main(String[] args) {
5          int n = Integer.parseInt(args[0]);
6          double total = 0.0;
7          for (int i = 1; i <= n; i++) {
8              total += 1.0 / i;
9          }
10         StdOut.println(total);
11     }
12 }
```

Variable Trace

line #	n	total	i
4			

```
>_
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i
5	3		

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i
6	3	0.0	

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i
7	3	0.0	1

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```
 1 import stdlib.StdOut;
 2
 3 public class Harmonic {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         double total = 0.0;
 7         for (int i = 1; i <= n; i++) {
 8             total += 1.0 / i;
 9         }
10         StdOut.println(total);
11     }
12 }
```

Variable Trace

line #	n	total	i
8	3	1.0	1

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```
 1  import stdlib.StdOut;
 2
 3  public class Harmonic {
 4      public static void main(String[] args) {
 5          int n = Integer.parseInt(args[0]);
 6          double total = 0.0;
 7          for (int i = 1; i <= n; i++) {
 8              total += 1.0 / i;
 9          }
10          StdOut.println(total);
11      }
12  }
```

Variable Trace

line #	n	total	i
7	3	1.0	2

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i
8	3	1.5	2

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
 1  import stdlib.StdOut;
 2
 3  public class Harmonic {
 4      public static void main(String[] args) {
 5          int n = Integer.parseInt(args[0]);
 6          double total = 0.0;
 7          for (int i = 1; i <= n; i++) {
 8              total += 1.0 / i;
 9          }
10          StdOut.println(total);
11      }
12  }
```

Variable Trace

line #	n	total	i
7	3	1.5	3

```
>_
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i
8	3	1.8333333333333333	3

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i
7	3	1.8333333333333333	4

```
$ java Harmonic 3
```

For Statements · Example (Harmonic Numbers)

```
 1 import stdlib.StdOut;
 2
 3 public class Harmonic {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         double total = 0.0;
 7         for (int i = 1; i <= n; i++) {
 8             total += 1.0 / i;
 9         }
10         StdOut.println(total);
11     }
12 }
```

Variable Trace

line #	n	total	i
10	3	1.8333333333333333	

```
$ java Harmonic 3
1.8333333333333333
```

For Statements · Example (Harmonic Numbers)

```java
import stdlib.StdOut;

public class Harmonic {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        double total = 0.0;
        for (int i = 1; i <= n; i++) {
            total += 1.0 / i;
        }
        StdOut.println(total);
    }
}
```

Variable Trace

line #	n	total	i

```
$ java Harmonic 3
1.8333333333333333
$ _
```

Break Statements

Break Statements

```
break;
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i

```
>_
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
1	1

>_

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
2	1

```
>_
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
3	1

```
>_
```

Break Statements

```
break;
```

Example

```java
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
6	1

>_

```
1
```

Break Statements

```
break;
```

Example

```java
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
7	2

>_

```
1
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
2	2

>_

```
1
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
3	2

>_

```
1
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
6	2

>_

```
1
2
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
7	3

>_

```
1
2
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
2	3

>_

```
1
2
```

Break Statements

```
break;
```

Example

```
1 int i = 1;
2 while (true) {
3     if (i >= 4) {
4         break;
5     }
6     StdOut.println(i);
7     i += 1;
8 }
```

Variable Trace

line #	i
3	3

>_

```
1
2
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
6	3

>_

```
1
2
3
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
7	4

```
>_
1
2
3
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
2	4

```
>_
```

```
1
2
3
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
3	4

>_

```
1
2
3
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i
4	4

>_

```
1
2
3
```

Break Statements

```
break;
```

Example

```
int i = 1;
while (true) {
    if (i >= 4) {
        break;
    }
    StdOut.println(i);
    i += 1;
}
```

Variable Trace

line #	i

```
>_
1
2
3
```

Continue Statements

Continue Statements

```
continue;
```

Continue Statements

```
continue;
```

Example

```
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
1	1

```
>_
```

Continue Statements

```
continue;
```

Example

```
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
2	1

Continue Statements

```java
continue;
```

Example

```java
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
3	1

>_

Continue Statements

```
continue;
```

Example

```
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
1	2

```
>_
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
2	2

>_

Continue Statements

```java
continue;
```

Example

```java
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
5	2

>_

```
2
```

Continue Statements

```
continue;
```

Example

```
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
1	3

```
2
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
2	3

```
>_
2
```

Continue Statements

```
continue;
```

Example

```
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
3	3

```
>_
2
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
1	4

```
>_
2
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
2	4

>_

```
2
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
5	4

```
>_
2
4
```

Continue Statements

```
continue;
```

Example

```
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
1	5

```
>_
2
4
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
2	5

```
>_
2
4
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
3	5

>_

```
2
4
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
1	6

```
>_
2
4
```

Continue Statements

```java
continue;
```

Example

```java
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
2	6

>_

```
2
4
```

Continue Statements

```java
continue;
```

Example

```java
1 for (int i = 1; i <= 6; i++) {
2     if (i % 2 != 0) {
3         continue;
4     }
5     StdOut.println(i);
6 }
```

Variable Trace

line #	i
5	6

>_

```
2
4
6
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i
1	7

>_

```
2
4
6
```

Continue Statements

```
continue;
```

Example

```
for (int i = 1; i <= 6; i++) {
    if (i % 2 != 0) {
        continue;
    }
    StdOut.println(i);
}
```

Variable Trace

line #	i

```
>_
2
4
6
```

Nesting

Nesting

If, while, and for statements can be nested within one another

Nesting · Example (Divisor Pattern)

Nesting · Example (Divisor Pattern)

DivisorPattern.java

Command-line input	n (int)
Standard output	a table where entry (i, j) is a star ("*") if j divides i or i divides j and a space (" ") otherwise

Nesting · Example (Divisor Pattern)

DivisorPattern.java

Command-line input	n (int)
Standard output	a table where entry (i, j) is a star ("*") if j divides i or i divides j and a space (" ") otherwise

~/workspace/dsaj

```
$ _
```

Nesting · Example (Divisor Pattern)

DivisorPattern.java

Command-line input	n (int)
Standard output	a table where entry (i, j) is a star ("*") if j divides i or i divides j and a space (" ") otherwise

~/workspace/dsaj

```
$ java DivisorPattern 5
```

Nesting · Example (Divisor Pattern)

DivisorPattern.java

Command-line input	*n* (int)
Standard output	a table where entry (i, j) is a star ("*") if j divides i or i divides j and a space (" ") otherwise

~/workspace/dsaj

```
$ java DivisorPattern 5
* * * * * 1
* *   *   2
*   *     3
* *   *   4
*       * 5
$ _
```

Nesting · Example (Divisor Pattern)

Nesting · Example (Divisor Pattern)

DivisorPattern.java

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Nesting · Example (Divisor Pattern)

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j

>_

```
$ _
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j

>_

```
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
1			

```
>_
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
3			

```
>_
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```java
 1  import stdlib.StdOut;
 2
 3  public class DivisorPattern {
 4      public static void main(String[] args) {
 5          int n = Integer.parseInt(args[0]);
 6          for (int i = 1; i <= n; i++) {
 7              for (int j = 1; j <= n; j++) {
 8                  if (i % j == 0 || j % i == 0) {
 9                      StdOut.print("* ");
10                  } else {
11                      StdOut.print("  ");
12                  }
13              }
14              StdOut.println(i);
15          }
16      }
17  }
```

Variable Trace

line #	n	i	j
4			

```
>_
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
5	3		

>_

```
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```java
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
6	3	1	

```
>_
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```
 1  import stdlib.StdOut;
 2
 3  public class DivisorPattern {
 4      public static void main(String[] args) {
 5          int n = Integer.parseInt(args[0]);
 6          for (int i = 1; i <= n; i++) {
 7              for (int j = 1; j <= n; j++) {
 8                  if (i % j == 0 || j % i == 0) {
 9                      StdOut.print("* ");
10                  } else {
11                      StdOut.print("  ");
12                  }
13              }
14              StdOut.println(i);
15          }
16      }
17  }
```

Variable Trace

line #	n	i	j
7	3	1	1

```
>_
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```java
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
8	3	1	1

```
>_
$ java DivisorPattern 3
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
9	3	1	1

```
$ java DivisorPattern 3
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	1	2

```
$ java DivisorPattern 3
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
8	3	1	2

```
$ java DivisorPattern 3
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
9	3	1	2

```
>_
$ java DivisorPattern 3
* *
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
7	3	1	3

>_

```
$ java DivisorPattern 3
* *
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
8	3	1	3

```
$ java DivisorPattern 3
* *
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
9	3	1	3

>_

```
$ java DivisorPattern 3
* * *
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	1	4

```
$ java DivisorPattern 3
* * *
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
14	3	1	

```
$ java DivisorPattern 3
* * * 1
```

Nesting · Example (Divisor Pattern)

```java
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
6	3	2	

```
$ java DivisorPattern 3
* * * 1
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	2	1

```
$ java DivisorPattern 3
* * * 1
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
8	3	2	1

>_

```
$ java DivisorPattern 3
* * * 1
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
9	3	2	1

```
$ java DivisorPattern 3
* * * 1
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	2	2

```
$ java DivisorPattern 3
* * * 1
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
8	3	2	2

>_

```
$ java DivisorPattern 3
* * * 1
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
9	3	2	2

>_

```
$ java DivisorPattern 3
* * * 1
* *
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	2	3

```
$ java DivisorPattern 3
* * * 1
* *
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
8	3	2	3

>_

```
$ java DivisorPattern 3
* * * 1
* *
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
11	3	2	3

```
$ java DivisorPattern 3
* * * 1
* *
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	2	4

>_

```
$ java DivisorPattern 3
* * * 1
* *
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
14	3	2	

```
$ java DivisorPattern 3
* * * 1
* *   2
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
6	3	3	

```
$ java DivisorPattern 3
* * * 1
* *   2
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	3	1

```
$ java DivisorPattern 3
* * * 1
* *   2
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
8	3	3	1

```
$ java DivisorPattern 3
* * * 1
* *   2
```

Nesting · Example (Divisor Pattern)

```java
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
9	3	3	1

```
>_
$ java DivisorPattern 3
* * * 1
* *   2
*
```

Nesting · Example (Divisor Pattern)

```
 1 import stdlib.StdOut;
 2
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
7	3	3	2

>_

```
$ java DivisorPattern 3
* * * 1
* *   2
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
8	3	3	2

```
$ java DivisorPattern 3
* * * 1
* *   2
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
11	3	3	2

```
$ java DivisorPattern 3
* * * 1
* *   2
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
7	3	3	3

```
$ java DivisorPattern 3
* * * 1
* *   2
*
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
8	3	3	3

```
$ java DivisorPattern 3
* * * 1
* *   2
*
```

Nesting · Example (Divisor Pattern)

```
 1  import stdlib.StdOut;
 2
 3  public class DivisorPattern {
 4      public static void main(String[] args) {
 5          int n = Integer.parseInt(args[0]);
 6          for (int i = 1; i <= n; i++) {
 7              for (int j = 1; j <= n; j++) {
 8                  if (i % j == 0 || j % i == 0) {
 9                      StdOut.print("* ");
10                  } else {
11                      StdOut.print("  ");
12                  }
13              }
14              StdOut.println(i);
15          }
16      }
17  }
```

Variable Trace

line #	n	i	j
9	3	3	3

```
>_
$ java DivisorPattern 3
* * * 1
* *   2
*   *
```

Nesting · Example (Divisor Pattern)

```java
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
7	3	3	4

>_

```
$ java DivisorPattern 3
* * * 1
* *   2
*   *
```

Nesting · Example (Divisor Pattern)

```java
 1 import stdlib.StdOut;
 2 
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable Trace

line #	n	i	j
14	3	3	

```
$ java DivisorPattern 3
* * * 1
* *   2
*   * 3
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j
6	3	4	

```
$ java DivisorPattern 3
* * * 1
* *   2
*   * 3
```

Nesting · Example (Divisor Pattern)

```java
import stdlib.StdOut;

public class DivisorPattern {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (i % j == 0 || j % i == 0) {
                    StdOut.print("* ");
                } else {
                    StdOut.print("  ");
                }
            }
            StdOut.println(i);
        }
    }
}
```

Variable Trace

line #	n	i	j

```
$ java DivisorPattern 3
* * * 1
* *   2
*   * 3
$ _
```

Variable Scope

Variable Scope

Part of the program that can refer to the variable by name

Variable Scope

Part of the program that can refer to the variable by name

Example

DivisorPattern.java

```java
 1 import stdlib.StdOut;
 2
 3 public class DivisorPattern {
 4     public static void main(String[] args) {
 5         int n = Integer.parseInt(args[0]);
 6         for (int i = 1; i <= n; i++) {
 7             for (int j = 1; j <= n; j++) {
 8                 if (i % j == 0 || j % i == 0) {
 9                     StdOut.print("* ");
10                 } else {
11                     StdOut.print("  ");
12                 }
13             }
14             StdOut.println(i);
15         }
16     }
17 }
```

Variable	Scope
`args`	lines 4 — 16
`n`	lines 5 — 16
`i`	lines 6 — 15
`j`	lines 7 — 13