CS 200 - Programming I: Loops

Marc Renault

Department of Computer Sciences
University of Wisconsin – Madison

Fall 2017
TopHat Sec 3 (PM) Join Code: 719946
TopHat Sec 4 (AM) Join Code: 891624
Loops
**While Loop**

```java
while (cond) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
**While Loop**

```java
while (cond) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
TopHat Question 1

What is the output?

```java
int i = 1;
while (i <= 5) {
    System.out.print(i + " ");
    i++;
}
```
Pre vs Post Incr/Decr Operators

Pre-Increment (resp. Decr)

++i
1. Increment i
2. Use i

What is the output?

```java
int i = 0;
System.out.print(++i);
System.out.print(i++);
System.out.print(i);
```
Pre vs Post Incr/Decr Operators

Pre-Increment (resp. Decr)

++i
1. Increment i
2. Use i

Post-Increment (resp. Decr)

i++
1. Use i
2. Increment i

TopHat Question 2

What is the output?

```java
int i = 0;
System.out.print(++i);
System.out.print(i++);
System.out.print(i);
```
## Pre vs Post Incr/Decr Operators

### Pre-Increment (resp. Decr)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
</table>
| ++i      | 1 Increment i  
|          | 2 Use i       |

### Post-Increment (resp. Decr)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
</table>
| i++      | 1 Use i     
|          | 2 Increment i |

### TopHat Question 2

What is the output?

```java
int i = 0;
System.out.print(++i);
System.out.print(i++);
System.out.print(i);
```
TopHat Question 3

What is the last number output?

```java
int i = 10;
while (i != 5) {
    System.out.print(i + " ");
    i -= 2;
}
```
Tracing a Loop

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }
    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```
Tracing a Loop

```java
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```
Tracing a Loop

```java
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }

}```
Tracing a Loop

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }
    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

Tracing a Loop

```
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>15</td>
<td>5</td>
</tr>
</tbody>
</table>

(Exit)
**Tracing a Loop**

1. **public class Sum {**
2. **public static int sum(int n) {**
3.     **int sum = 0;**
4.     **int i = 1;**
5.     **while (i <= n) {**
6.         **sum += i++;**
7.     **}**
8.     **return sum;**
9. **}**
10. **}**
11. 
12. **public static void main(String[] args) {**
13.     **System.out.print(sum(5));**
14. **}**
15. **}**

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Tracing a Loop

public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }
    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
**Tracing a Loop**

```
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Tracing a Loop

```
1 public class Sum {
2     public static int sum(int n) {
3         int sum = 0;
4         int i = 1;
5         while (i <= n) {
6             sum += i ++;
7         }
8         return sum;
9     }
10 }
11
12 public static void main(String[] args) {
13     System.out.print(sum(5));
14 }
15
16 }
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
**Tracing a Loop**

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
Tracing a Loop

```java
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }

}```
Tracing a Loop

```
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
Tracing a Loop

public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }

}
Tracing a Loop

```java
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

5/25
Tracing a Loop

```
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }
    public static void main(String[] args) {
        System.out.println(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
Tracing a Loop

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }
    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
Tracing a Loop

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```
Tracing a Loop

```
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>
Tracing a Loop

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.println(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>
Tracing a Loop

```
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>
Tracing a Loop

```java
public class Sum {

    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.println(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>
Tracing a Loop

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i;
            i = i + 1;
        }
        return sum;
    }
    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5 (Exit)</td>
<td>5</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>
Tracing a Loop

```java
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5 (Exit)</td>
<td>5</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>
Tracing a Loop

```
public class Sum {
    public static int sum(int n) {
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum += i ++;
        }
        return sum;
    }

    public static void main(String[] args) {
        System.out.print(sum(5));
    }
}
```

<table>
<thead>
<tr>
<th>Iteration</th>
<th>n</th>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>5 (Exit)</td>
<td>5</td>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>

\[ \frac{5}{25} \]
TopHat Question 4

What input causes the loop to stop?

Scanner sc = new Scanner(System.in);
char c = '\0';
while (c != 'n') {
    System.out.println("This is the song that never ends");
    System.out.println("It goes on and on my friends");
    System.out.println("Some people started singing it, not knowing what it was");
    System.out.println("And they’ll continue singing it forever just because");
    c = sc.next().charAt(0);
}
For Loop

```java
for (init; cond; incr) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
For Loop

```
for (init; cond; incr) {
    loopStatement1;
    loopStatement2;
    ...
    loopStatementN;
}
```
TopHat Question 5

What is the output?

```java
for (int i = 1; i <= 5; ++i) {
    System.out.print(i + " ");
}
```
TopHat Question 6

What is the output?

```java
for (int i = 0, j = 1, k = 3; i < 3; i++, j += 2, k += 3) {
    System.out.print(i + " " + j + " " + k + " ");
}
```
For vs While

```c
for (init; cond; incr) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```

```c
    {
        init;
        while (cond) {
            loopStatement1;
            loopStatement2;
            .
            .
            .
            loopStatementN;
            incr;
        }
    }
```
For vs While

```java
for (init; cond; incr) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```

```java`
while (cond) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
    incr;
}
```

For Notes
- `init` – Scope limited to the for loop.
- All 3 of init, cond and incr are optional.
TopHat Question 7

Replace the ????? so that the for loop calculates the same sum as the while.

```java
public static int sum(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}
```

```java
public static int sum(int n) {
    int sum = 0;
    int i = 1;
    while (i <= n) {
        sum += i++;
    }
    return sum;
}
```
Do-While Loop

do {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
} while (cond);
**Do-While Loop**

```java
do {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
} while (cond);
```

Diagram:
- **Control Flow**
  - **Loop Statement Block**
  - **cond**
    - **true**
    - **false**
  - **Following Statements**
**TopHat Question 8**

What is the output?

```java
int i = 1;
do {
    System.out.print(i + " ");
i++;
} while (i <= 5);
```
Picking a Loop

**while Loop**

When the number of iterations cannot be calculated in advance of entering the loop. E.g. depends on user input.
<table>
<thead>
<tr>
<th>Loop Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>while Loop</strong></td>
<td>When the number of iterations cannot be calculated in advance of entering the loop. E.g. depends on user input.</td>
</tr>
<tr>
<td><strong>for Loop</strong></td>
<td>When the number of iterations can be calculated in advance of entering the loop. E.g. Summing from 1 to n.</td>
</tr>
</tbody>
</table>
Picking a Loop

**while Loop**
When the number of iterations cannot be calculated in advance of entering the loop. E.g. depends on user input.

**for Loop**
When the number of iterations can be calculated in advance of entering the loop. E.g. Summing from 1 to n.

**do-while Loop**
Same as the while, excepted that the loop must be executed at least once. E.g. a program menu.
Palindrome Exercise

Write a method that determines if a string is a palindrome.
**Branching Statements**

Stopping the Loop

- `break` – Exits the loop
Branching Statements

Stopping the Loop

- **break** – Exits the loop
- **return** – Returns a value and exits the current method
Branching Statements

Stopping the Loop
- `break` – Exits the loop
- `return` – Returns a value and exits the current method

Skipping to the next iteration
- `continue` – Jumps to the next iteration
  Note that, in a `for` loop, this jumps to the increment statement.
TopHat Question 9

What is the output?

```java
for (int i = 1; i <= 10; ++i) {
    if (i > 5) break;
    if (i % 2 == 0) continue;
    System.out.print(i + " ");
}
```
TopHat Question 10

What is the output?

```java
int j = 1;
int i = 1;
while (i <= 10) {
    if (j++ > 5) break;
    if (i % 2 == 0) continue;
    System.out.print(i++ + " ");
}
```
**For Loop**

```plaintext
for (init; cond; incr) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
For Loop

```java
for ( ; cond; incr) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
**For Loop**

```java
for ( ; cond; incr) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
**For Loop**

```java
for (init; cond; ) {
    loopStatement1;
    loopStatement2;
    ...
    loopStatementN;
}
```
For Loop

```java
for (init; cond; ) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
**For Loop**

```java
for ( ; cond; ) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
For Loop

```java
for ( ; cond; ) {
    loopStatement1;
    loopStatement2;
    .
    .
    .
    loopStatementN;
}
```
For Loop

```java
for (init; ; incr) {
    loopStatement1;
    loopStatement2;
    ...
    loopStatementN;
}
```
For Loop

```java
for (init ; ; incr) {
    loopStatement1;
    loopStatement2;
    ...
    loopStatementN;
}
```
Infinite Loops

- The compiler does not check for infinite loops.
Infinite Loops

- The compiler does not check for infinite loops.
- Does not cause runtime errors.
Infinite Loops

- The compiler does not check for infinite loops.
- Does not cause runtime errors.
- There are situations where programmers use infinite loops, such as:
Infinite Loops

The compiler does not check for infinite loops.
Does not cause runtime errors.
There are situations where programmers use infinite loops, such as:
- Graphical user interfaces
Infinite Loops

- The compiler does not check for infinite loops.
- Does not cause runtime errors.
- There are situations where programmers use infinite loops, such as:
  - Graphical user interfaces
  - Server applications
**Infinite Loops**

- The compiler does not check for infinite loops.
- Does not cause runtime errors.
- There are situations where programmers use infinite loops, such as:
  - Graphical user interfaces
  - Server applications
  - Services / daemons
Infinite Loops

The compiler does not check for infinite loops.
Does not cause runtime errors.
There are situations where programmers use infinite loops, such as:
- Graphical user interfaces
- Server applications
- Services / daemons

Preferred syntax:

```java
while (true) {
    .
    .
    .
}
```

```java
for ( ; ; ) {
    .
    .
    .
}
```
Nested Loops

- Multi-dimensional problems.
- Repetitive tasks for every iteration.
Nested Loops

- Multi-dimensional problems.
- Repetitive tasks for every iteration.

Branching Statements
break and continue affect only the innermost loop in which they are contained.
TopHat Question 11

What is the output?

```java
for (int i = 1; i <= 5; ++i) {
    System.out.print("\n" + i + ": ");
    for (int j = i; j <= 10; j++) {
        if (i % 2 == 0) continue;
        if (j % 3 == 0) break;
        System.out.print(j + " ");
    }
}
```
Enumerations
Enumerations Basics

- An *enumeration* type is a variable type with values restricted to those listed in the enumeration.
- Used for grouping related constants.
- Works in a switch statement.
Enumerations

```java
public enum EnumIdentifier {ENUM1, ENUM2, ...}
```

### Enumerations Basics

- **An enumeration** type is a variable type with values restricted to those listed in the enumeration.
- **Used for grouping related constants.**
- **Works in a switch statement.**

```java
class EnumEx {
    public enum AnimalType {DOG, CAT, FISH, COW}

    public static String speak(AnimalType a) {
        String toRet = "";
        switch (a) {
        case DOG:
            toRet = "WOOF!";
            break;
        case CAT:
            toRet = "MEOW!";
            break;
        case FISH:
            toRet = "GLUB!";
            break;
        case COW:
            toRet = "MOO!";
            break;
        default:
            toRet = "";
            break;
        }
        return toRet;
    }

    public static void main(String[] args) {
        System.out.print(speak(AnimalType.FISH));
    }
}
```
Debugging
DEBUGGING

What is a bug?

- A *bug* in an error in a program that causes undesirable behaviour.
- Origins $^a$:
  - Classic story: Sept 9, 1945, a moth in a computer at Harvard was the cause of unexpected behaviour.

---

What is a bug?

- A *bug* in an error in a program that causes undesirable behaviour.
- Origins:
  - Classic story: Sept 9, 1945, a moth in a computer at Harvard was the cause of unexpected behaviour.
  - Term actually used as early as 1878 by Thomas Edison.

---

**Debugging**

What is a bug?

- A *bug* in an error in a program that causes undesirable behaviour.
- Origins:\n  - Classic story: Sept 9, 1945, a moth in a computer at Harvard was the cause of unexpected behaviour.
  - Term actually used as early as 1878 by Thomas Edison.
  - Etymology relates to bugaboo or bugbear.

---

Debugging

What is a bug?

- A bug in an error in a program that causes undesirable behaviour.
- Origins:
  - Classic story: Sept 9, 1945, a moth in a computer at Harvard was the cause of unexpected behaviour.
  - Term actually used as early as 1878 by Thomas Edison.
  - Etymology relates to bugaboo or bugbear.

---

Eclipse Debugger

Eclipse provides a debugger that can be very helpful to trace the code and find those bugs.

FURTHER READING

COMP SCI 200: Programming I
zyBook code:
WISCCOMPSCI200Fall2017

- Chapter 6. Loops
Appendix
Image Sources I

https://brand.wisc.edu/web/logos/

http://www.zybooks.com/