CSE115 / CSE503
Introduction to Computer Science I

Dr. Carl Alphonce
343 Davis Hall
alphonce@buffalo.edu

Office hours:
Thursday 12:00 PM – 2:00 PM
Friday 8:30 AM – 10:30 AM
OR request appointment via e-mail
Turn off and put away electronics:

- cell phones
- pagers
- laptops
- tablets
- etc.
ROADMAP
Where we’ve been
  Interfaces
  event-driven programming

Today
  control structures

Where we’re heading
  collections
  search
I have unclaimed exams with me.

Pick up at end of class.

Grading questions? Come to office hours.
channel 1
What do methods in an interface lack that methods in a class have?

<table>
<thead>
<tr>
<th>Rank</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BODY</td>
</tr>
<tr>
<td>2</td>
<td>INSTANTIATION</td>
</tr>
<tr>
<td>3</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>4</td>
<td>DEFINITIONS</td>
</tr>
<tr>
<td>5</td>
<td>CONSTRUCTOR</td>
</tr>
</tbody>
</table>

Keyword: code, body, ...

Keyword Matches: 35
### What do methods in an interface lack that methods in a class have?

<table>
<thead>
<tr>
<th>Rank</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Other</td>
</tr>
</tbody>
</table>

Convince your neighbor your answer is correct.
What do methods in an interface lack that methods in a class have?

<table>
<thead>
<tr>
<th>Rank</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BODY</td>
</tr>
<tr>
<td>2</td>
<td>INSTANTIATION</td>
</tr>
<tr>
<td>3</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>4</td>
<td>DEFINITIONS</td>
</tr>
</tbody>
</table>

Keyword: code, body, ...

Keyword Matches: 141
CONTROL STRUCTURES
Recall that not every language is a programming language.

A programming language must be able to express:
- sequence
- selection
- repetition

In Java:
- sequence can be expressed by the order of statements and the nesting of expressions
- selection can be expressed by conditional statements (if statement, if-else statement), as well as by polymorphism
- repetition can be expressed by looping statements (while statement, for statement, foreach statement)

There are additional ways in which selection and repetition can be expressed (e.g. pattern matching, unification / recursion)
Control structures

conditional statements (SELECTION)
  if-else statement
  if statement

loop statements (REPETITION)
  while statement (while loop)
  for statement (for loop)
Control Structures

→ selection

→ if-else statement
if ( <expr> ) <stmt₁> else <stmt₂>
Control Structures

- selection

- if statement
if ( <expr> ) <stmt>

- If the expression is true, execute the statement.
- If the expression is false, do nothing.
The boolean type has two values, **true** and **false**.

### Boolean Operators

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P &amp;&amp; Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

| P   | Q   | P || Q |
|-----|-----|--------|
| true| true| true   |
| true| false| true   |
| false| true| true   |
| false| false| false  |

<table>
<thead>
<tr>
<th>P</th>
<th>!P</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
relational operators:
   less than: <
   less than or equal to: <=
   equal to: ==
   not equal to: !=
   greater than or equal to: >=
   greater than: >

relational operators: $D \times D \rightarrow \text{boolean}$, where $D$ is a numeric primitive type (e.g. int, double)
channel 1
What value does the following method call return?

foo(8)

```java
public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}
```

A. Fabulous
B. Tremendous
C. Wicked
D. all of the above
E. none of the above
What value does the following method call return? foo(8)

public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}

A. Fabulous
B. Tremendous
C. Wicked
D. all of the above
E. none of the above
What value does the following method call return?

foo(8)

public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}

A. Fabulous
B. Tremendous
C. Wicked ✔
D. all of the above
E. none of the above
What value does the following method call return?

foo(8)

```java
public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}
```

A. Fabulous  
B. Tremendous  
C. Wicked  
D. all of the above  
E. none of the above
channel 1
What is the largest set of argument values for which the method returns “Wicked” on all values?

A. 4, 5, 6, 7  
B. 4, 5, 6  
C. 5, 6, 7  
D. 5, 6  
E. 5
What is the largest set of argument values for which the method returns “Wicked” on all values?

public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}

A. 4, 5, 6, 7
B. 4, 5, 6
C. 5, 6, 7
D. 5, 6
E. 5
What is the largest set of argument values for which the method returns “Wicked” on all values?

```java
public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}
```

A. 4, 5, 6, 7
B. 4, 5, 6
C. 5, 6, 7
D. 5, 6
E. 5
What is the largest set of argument values for which the method returns “Wicked” on all values?

public String foo(int x) {
    if (x <= 4) { return "Fabulous"; }
    if (x > 6) { return "Tremendous"; }
    return "Wicked";
}

A. 4, 5, 6, 7
B. 4, 5, 6
C. 5, 6, 7
D. 5, 6
E. 5
Control Structures

→ repetition

→ while statement
while ( <expr> ) <stmt>
while ( <expr> ) <stmt>
if ( <expr> ) <stmt>
Recall the flow chart for an if statement: if `<expr>` is true, then `<stmt>` is executed once. Afterwards execution continues with the next statement in the program.
**while** ( <expr> ) <stmt>

In the case that <expr> is true, <stmt> is executed. Control goes back and re-evaluates <expr>, thus creating a loop in the flow of control. In the case that <expr> is false the loop is not entered and program execution continues with the next statement.

The flow chart for a **while statement** is different from that of the **if statement** in one very important respect.
public void message() {
    System.out.print("I ");
    int timesAlreadyPrinted = 0;
    while (timesAlreadyPrinted < 3) {
        System.out.print("really ");
        timesAlreadyPrinted = timesAlreadyPrinted + 1;
    }
    System.out.println(" like spring break!");
}
We spent some time tracing the execution of the code on the previous slide, keeping track of the value of the variable `timesAlreadyPrinted` and the corresponding output.

Being able to trace the execution of code by hand is an important skill.