Objects and Classes
(Part 1)

Introduction to Programming and Computational Problem Solving - 2
CSE 8B
Lecture 6
Announcements

• Assignment 2 is due today, 11:59 PM
• Quiz 2 is Oct 23
• Assignment 3 will be released today
  – Due Oct 28, 11:59 PM
• Educational research study
  – Oct 23, weekly reflection
• Reading
  – Chapter 9
Object-oriented programming

• Object-oriented programming (OOP) involves programming using objects

• This is the focus of CSE 8B
  – The previous four lectures have been “double speed”
  – Beginning with this lecture, they will be “half speed”
Objects and classes

• An object represents an entity in the real world that can be distinctly identified
  – For example, a student, a desk, a circle, a button, and even a loan can all be viewed as objects
  – An object has a unique identity, state, and behaviors

• Classes are constructs that define objects of the same type
Objects

• An object has a unique identity, state, and behaviors

• The state of an object consists of a set of data fields (also known as properties) with their current values

• The behavior of an object is defined by a set of methods
Objects

• An object has both a state and behavior
  – The state defines the object
  – The behavior defines what the object does

Class Name: Circle
Data Fields:
  radius is _______
Methods:
getArea

Circle Object 1
Data Fields:
  radius is 10

Circle Object 2
Data Fields:
  radius is 25

Circle Object 3
Data Fields:
  radius is 125

A class template

Three objects of the Circle class
Classes

• A Java class uses variables to define data fields and methods to define behaviors
• Additionally, a class provides a special type of methods, known as constructors, which are invoked to construct objects from the class
class Circle {
    /** The radius of this circle */
    double radius = 1.0;

    /** Construct a circle object */
    Circle() {
    }

    /** Construct a circle object */
    Circle(double newRadius) {
        radius = newRadius;
    }

    /** Return the area of this circle */
    double getArea() {
        return radius * radius * 3.14159;
    }
}
### UML Class Diagram

<table>
<thead>
<tr>
<th>Class</th>
<th>Data Fields</th>
<th>Constructors and Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>radius: double</td>
<td>Circle()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Circle(newRadius: double)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getArea(): double</td>
</tr>
<tr>
<td></td>
<td></td>
<td>getPerimeter(): double</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setRadius(newRadius: double): void</td>
</tr>
</tbody>
</table>

- **circle1**: Circle
  - radius = 1.0

- **circle2**: Circle
  - radius = 25

- **circle3**: Circle
  - radius = 125

---

**UML notation for objects**
Constructors

• Constructors must have the same name as the class itself
• A constructor with no parameters is referred to as a no-arg constructor
• Constructors do not have a return type
  – Not even void
• Constructors are invoked using the new operator when an object is created
• Constructors play the role of initializing objects
Creating objects using constructors

new ClassName();

• For example
  new Circle();
  new Circle(5.0);
Default constructor

• A class may be defined without constructors
• In this case, a no-arg constructor with an empty body is implicitly defined in the class
• This constructor, called a *default constructor*, is provided automatically only if no constructors are explicitly defined in the class
Declaring object reference variables

• To reference an object, assign the object to a reference variable
• To declare a reference variable, use the syntax `ClassName objectRefVar;`
• For example
  `Circle myCircle;`
Declaring and creating in one step

ClassName objectRefVar = new ClassName();

For example
Circle myCircle = new Circle();
Accessing an object’s members

• Use the *object member access operator*
  – Also called the *dot operator* (.)

• Reference the object’s data using
  `objectRefVar.data`
  – For example
    `myCircle.radius`

• Invoke the object’s method using
  `objectRefVar.methodName(arguments)`
  – For example
    `myCircle.getArea()`
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;

Create a new Circle object

: Circle
radius: 5.0
Trace code

Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;

Assign object reference to myCircle

myCircle reference value

: Circle
radius: 5.0
Circle myCircle = new Circle(5.0);

Circle yourCircle = new Circle();

yourCircle.radius = 100;
Trace code

Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;

myCircle
reference value

: Circle
radius: 5.0

yourCircle
no value

: Circle
radius: 1.0

Create a new Circle object
Trace code

Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();
yourCircle.radius = 100;

Assign object reference to yourCircle

myCircle reference value

: Circle
radius: 5.0

yourCircle reference value

: Circle
radius: 1.0
Circle myCircle = new Circle(5.0);
Circle yourCircle = new Circle();

yourCircle.radius = 100;

Change radius in yourCircle
Reference data fields and null

- The data fields can be of reference types
- For example, the following Student class contains a data field name of the String type

```java
public class Student {
    String name;
    int age;
    boolean isScienceMajor;
    char gender;
}
```

- If a data field of a reference type does not reference any object, then the data field holds the special Java literal value `null`
Default value for a data field

• The default value of a data field is null for a reference type
  θ for a numeric type
  false for a boolean type
  '\u0000' for a char type

```java
public class Student {
    String name; // name has default value null
    int age; // age has default value 0
    boolean isScienceMajor; // isScienceMajor has default value false
    char gender; // c has default value '\u0000'
}
```
Default values

• **Note:** Java assigns no default value to a local variable inside a method

```java
public class Test {
    public static void main(String[] args) {
        int x; // x has no default value
        String y; // y has no default value
        System.out.println("x is "+ x);
        System.out.println("y is "+ y);
    }
}
```

Compile error: variable not initialized
Differences between variables of primitive data types and object types

• A variable of a primitive type holds a value of the primitive type

• A variable of a reference type holds a reference to where an object is stored in memory

<table>
<thead>
<tr>
<th>Primitive type</th>
<th>int i = 1</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object type</td>
<td>Circle c</td>
<td>c</td>
</tr>
</tbody>
</table>

Created using new Circle()

- c: Circle
- radius = 1
Differences between variables of primitive data types and object types

- **Variable assignment**

  **Primitive type assignment**  \( i = j \)

  - **Before:**
    - \( i \) = 1
    - \( j \) = 2

  - **After:**
    - \( i \) = 2
    - \( j \) = 2

  **Object type assignment**  \( c1 = c2 \)

  - **Before:**
    - \( c1 \)
    - \( c2 \)
    - \( c1: \text{Circle} \\
      \text{radius} = 5 \\
    
    - \( c2: \text{Circle} \\
      \text{radius} = 9 \)

  - **After:**
    - \( c1 \)
    - \( c2 \)
    - Crossed out
    - \( c1: \text{Circle} \\
      \text{radius} = 5 \\
    
    - \( c2: \text{Circle} \\
      \text{radius} = 9 \)
Using classes from the Java library

• The Java API contains a rich set of classes for developing Java programs
• Some commonly used ones
  – The String class
  – The java.util.Date class
  – The Math class
  – The java.util.Random class
    • More capable than Math.random method
Instance methods vs static methods

• An instance method can only be invoked from a specific instance of an object
  – The syntax to invoke an instance method is `referenceVariable.methodName(arguments)`

• A static method (i.e., a non-instance method) can be invoked without using an object (i.e., they are not tied to a specific object instance)
  – The syntax to invoke a static method is `ClassName.methodName(arguments)`
Next Lecture

• Objects and classes
• Reading
  – Chapter 9