CISC 3115 MY3

Defining Java Class

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- Defining classes for objects

- UML class diagram

- Accessing objects via reference variables

Recall: Authoring a Java Program

- Let's consider the following 5 components

  - Requirement

  - Design

  - Implementation

  - Verification (commonly, testing)

  - Validation

- Call them 5 components instead of 5 steps, because it is not necessary to follow them in the above order

Recall: Requirements

- About answering question:
  - What does the "customer" want? Call the answer the requirement.
    - In the class:
      - What does the instructor want?
    - For your own exploration:
      - What do I want?

- Programmers provide a technical solutions in the means of software/programs to customers

- Is what we learned sufficient?

Review: What Have We Learned for Java Programming?

Review: What We Have Learned for Java Programming

- A divide-and-conquer problem solving strategy

  - Divide the problem into smaller problem, and solve each using a Java method (or function)

- Java programming language

Review: What We Have Learned?

- Functions (In Java, we call them methods)
  - Data types and variables
  - Arrays
  - Statements
  - Comments
  - Flow control
    - Selections
    - Iterations/Loops
  - …

```
class PlayNumbers {

    public static void main(String[] args) {

    }

    public static int addTwo(int a, int b) {

        return a + b;

    }

    public static boolean isPositive(int a) {

        if (a > 0) return true;

        else return false;

    }

    // ……
}
```

Review: Arrays and *Something New*?

- Examine the code,
  - What data type is "String[] args" in the main method's parameter list?
  - Can we print it out?
  - But who calls main and passes the argument to it?

- Something new?
  - Command line arguments

```
class PlayNumbers {

    public static void main(String[] args) {

    }

    public static int addTwo(int a, int b) {

        return a + b;

    }

    public static boolean isPositive(int a) {

        if (a > 0) return true;

        else return false;

    }

    // ……

}
```

Questions

- Have you been working on CodeLab assignments?

- Have you been reviewing topics in CISC 1115?

- How to use the command line arguments?

From Class to Objects

```
class PlayNumbers {

    public static void main(String[] args) {

    }

    public static int addTwo(int a, int b) {

        return a + b;

    }

    public static boolean isPositive(int a) {

        if (a > 0) return true;

        else return false;

    }

    // ……

}
```

```
class PlayNumbersToo {

    private int op1, op2;

    public static void main(String[] args) {

    }

    public int addTwo(int a, int b) {

        return a + b;

    }

    public boolean isPositive(int a) {

        if (a > 0) return true;

        else return false;

    }

    // ……

}
```

But, what's the benefit?

Object-Oriented Programming

- Problem solving and programming using objects
  - which enables a divide-and-conquer problem solving strategy using multiple classes and objects and via object-oriented "modeling"
- An *object* represents an entity in the real world that can be distinctly identified.
  - Student, instructor, class
  - Building, room, desk
  - Circle, rectangle
  - Button, menu
  - Loan, sales transaction
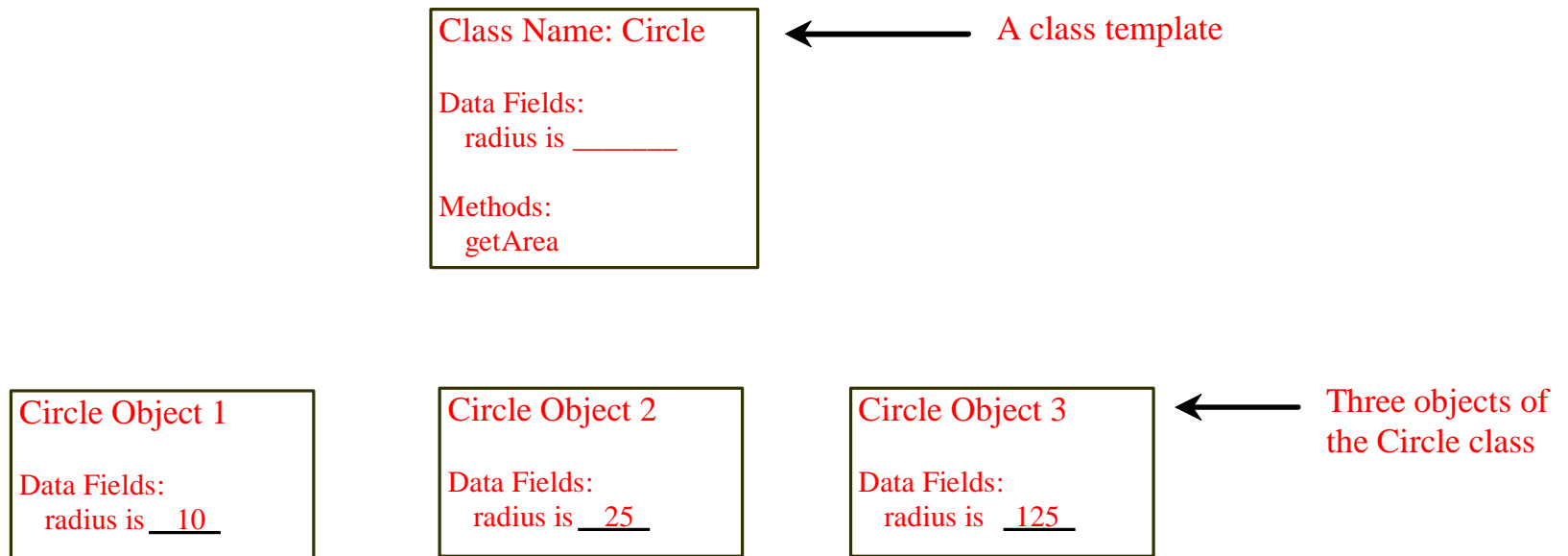
Object, State, and Behavior

- 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 representing what it does.

Classes

- Define objects of the same type, a template that an object can be created from.

- 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.

Objects and Classes

- From a class, we can create objects of the class

Class Name: Circle

Data Fields:
  radius is _____

Methods:
  getArea

← A class template

Circle Object 1

Data Fields:
  radius is __10__

Circle Object 2

Data Fields:
  radius is __25__

Circle Object 3

Data Fields:
  radius is __125__

← Three objects of the Circle class

Objects and Classes: State and Behavior

- A Java class uses variables to define data fields and methods to define behaviors.

  - The state of an object of the class corresponds to the data fields and their values.

  - The behavior of the object corresponds to the methods.

    - Constructors

      - A special type of methods that are invoked to initialize the data fields when the object is being constructed.

A Circle Class

```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;                        ←  Data field

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

  /** Return the area of this circle */
  double getArea() {                          ←  Method
    return radius * radius * 3.14159;
  }
}
```

More discussion about these in next class

Writing the Circle Class


```
hui@ThinkpadE450 MINGW64 ~/work/
e (master)
$ atom Circle.java

hui@ThinkpadE450 MINGW64 ~/work/
e (master)
$ |
```

Circle.java — C:\Users\hui\work\course\CISC3115\SamplePrograms\DefineCl

File   Edit   View   Selection   Find   Packages   Help

Circle
  Circle.java

Circle.java

```java
1    class Circle {
2      double radius = 1;
3
4      Circle() {
5      }
6
7      Circle(double newRadius) {
8        radius = newRadius;
9      }
10
11     double getArea() {
12       return radius * radius * Math.PI;
13     }
14
15     double getPerimeter() {
16       return 2 * radius * Math.PI;
17     }
18
19     void setRadius(double newRadius) {
20       radius = newRadius;
21     }
22   }
23
```
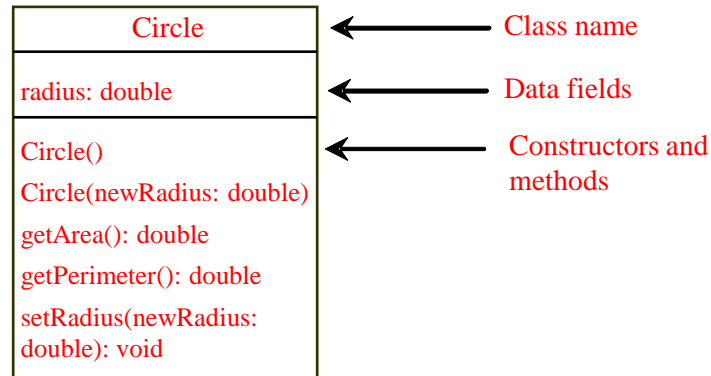
Reading the Circle Class

- Compared to the programs you written, is there any notable difference?

    - Does it have a main method?

    - Can you run it?

    - Can you compile it?

    - Is there any constructors? Where are they? How are constructors named? Does a constructor have a return type?

    - Where are the methods that define the behavior an object created from the class? Must a method have a return type? What are the return types?

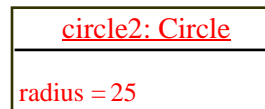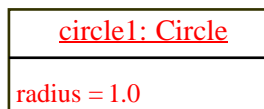   - Can method take a parameter? Must a parameter have a type and name?

Representing Class and Objects in UML Diagram

- UML = Unified Modeling Language

UML Class Diagram

Circle
radius: double
Circle() Circle(newRadius: double) getArea(): double getPerimeter(): double setRadius(newRadius: double): void

← Class name

← Data fields

← Constructors and methods

UML notation for objects

circle1: Circle
radius = 1.0

circle2: Circle
radius = 25

circle3: Circle
radius = 125
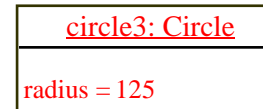
Reading the UML Diagram

- How does a class diagram depict a class?

  - How is a data field presented?

  - How is a constructor represented?

  - How is a method represented?

- How is an object represented in UML?

Observations

- We can compile the Circle class, but we cannot run it. Why can we not run it?

- The Circle class acts as a template from which objects of the Circle class can be created, but have we created any objects from the Circle class?

The TestCircle Class

- It has a main method

  - A number of Circle objects are created.

  - The areas of the Circle objects are computed and printed out

Writing the TestCircle Class

MINGW64:/c/Users/hui/work/course/CISC3115/SamplePrograms/DefineClass/Circle

```
Circle.class   Circle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circl
e (master)
$ atom TestCircle.java
```

```java
TestCircle.java

1   class TestCircle {
2       public static void main(String[] args) {
3           Circle c1 = new Circle();
4           System.out.println("The area of the circle of radius " + c1.getRadius() + " is " + c1.getArea());
5
6           Circle c2 = new Circle(25);
7           System.out.println("The area of the circle of radius " + c2.getRadius() + " is " + c2.getArea());
8
9           Circle c3 = new Circle(125);
10          System.out.println("The area of the circle of radius " + c3.getRadius() + " is " + c3.getArea());
11      }
12  }
13
```

Compiling and Running the Program

CUNY | Brooklyn College

Setters and Getters

- Getters are methods to return the value of a data field of an object

- Setters are methods to change the value of a data field of an object

```
class Circle {
// …
    public void setRadius(double r) {
        this.r = r;
    }
    public void getRadius() {
        return r;
    }
//…
}
```

CUNY | Brooklyn College

Accessing objects via reference variables

- Using classes with instance methods and varaibles

  - Create objects from classes

  - Access objects via reference variables

Example:

      Circle circle = new Circle(25);

      area = circle.getArea();

Where circle is a reference variable that references a Circle object

Static (Class) vs. Instance Variables and Methods

- Instance variables and instance methods defined in a class, but are part of an object (not the class).

- static variables and methods defined in a class, and also part of the class.

Given

```
class A {
    public static void m1() {
    }

    public void m2() {
    }
}
```

Correct or Wrong?

```
class B {
    public void m1() {
        A.m1();
        A.m2();
    }
}
```

In-Class Exercise 1

- Write the Circle and TestCircle classes

- Compile and run the program

- Have free time? Go to next slide

- The instructor will observe demos of the program from randomly selected students.

In-Class Exercise 2

- Write two classes, TV and TestTV as illustrated in Listings 9.3 and 9.4 in the textbook

- Compile and run the program

Questions

- Concepts of objects and classes

- Relationship between objects and classes

- Defining class in Java

- Depicting class in UML

- Creating and referencing objects