

CISC 3115 TY2

# Defining Java Class

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

# Notice

- The slides are always subject to change.
- The slides posted before the lecture are for preview only, and they are a draft and their content can change significantly.

# Outline

- Review
- Defining classes for objects
- UML class diagram
- Accessing objects via reference variables
- Primitive and 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?

- 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?
  - 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?



# To Discuss: 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;  
    }  
    // .....  
}
```

# Object-Oriented Programming

- Programming using objects
- 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

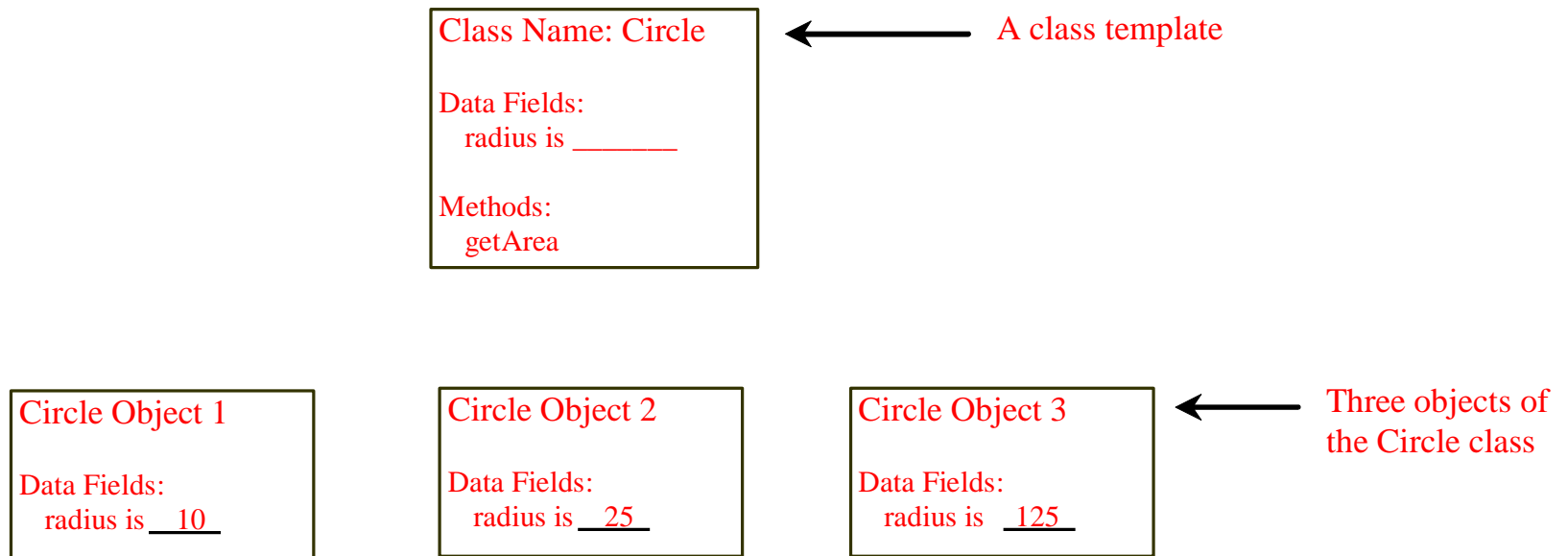
- An object has a unique identity, state, and behaviors.
  - The *state* of an object consists of a set of *data fields* (also known as *properties*) with their current values.
  - The *behavior* of an object is defined by a set of methods representing what it does.

# 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



# 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;   
  
    /** 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;  
    }  
}
```

**Data field**

**Constructors**

**Method**

More  
in  
Next  
class

# Writing the Circle Class

```
MINGW64:/c:/Users/hui/work/course/CISC3115/Samp Circle.java — C:\Users\hui\work\course\CISC3115\SamplePrograms\DefineCI
File Edit View Selection Find Packages Help
Circle
  Circle.java
hui@ThinkpadE450 MINGW64 ~/work/
e (master)
$ atom Circle.java
hui@ThinkpadE450 MINGW64 ~/work/
e (master)
$ |

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
```



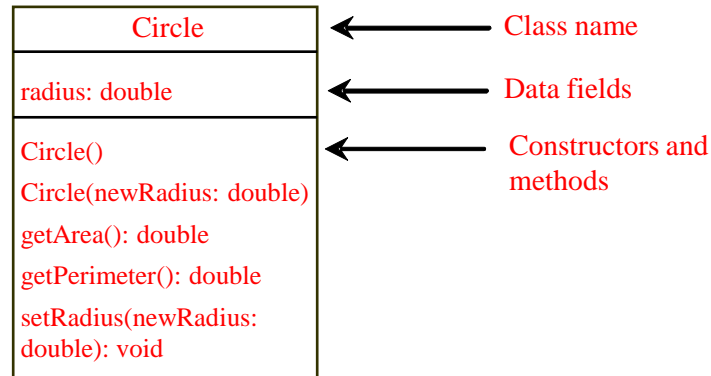
# Reading the Circle Class

- 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?

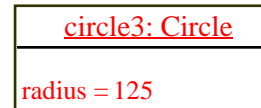
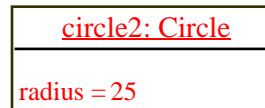
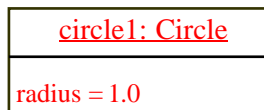
# Representing Class and Objects in UML Diagram

- UML = Unified Modeling Language

UML Class Diagram



UML notation for objects



# 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?

# Questions

- Concepts of objects and classes
- Relationship between objects and classes
- Defining class in Java
- Depicting class 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/Circle
e (master)
$ atom TestCircle.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

```
MINGW64:/c:/Users/hui/work/course/CISC3115/SamplePrograms/DefineClass/Circle
hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circle
e (master)
$ ls
Circle.class  Circle.java  TestCircle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circle
e (master)
$ javac TestCircle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circle
e (master)
$ ls
Circle.class  Circle.java  TestCircle.class  TestCircle.java

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circle
e (master)
$ java TestCircle
The area of the circle of radius 1.0 is 3.141592653589793
The area of the circle of radius 25.0 is 1963.4954084936207
The area of the circle of radius 125.0 is 49087.385212340516

hui@ThinkpadE450 MINGW64 ~/work/course/CISC3115/SamplePrograms/DefineClass/Circle
e (master)
```



# Static (Class) vs. Instance

- 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() {  
    }  
}
```

## What about this? (correct or wrong)

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

# 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;  
    }  
    //...  
}
```

# Exercise (in Groups)

- Write the Circle and TestCircle classes
- Compile and run the program
- Submit your work as a journal entry
- Have free time? Go to next slide

# Exercise (in Groups)

- Write two classes, TV and TestTV as illustrated in Listings 9.3 and 9.4 in the textbook
- Compile and run the program
- Submit your work as a journal entry