CISC 3115

Constructing and Referencing Objects

Hui Chen

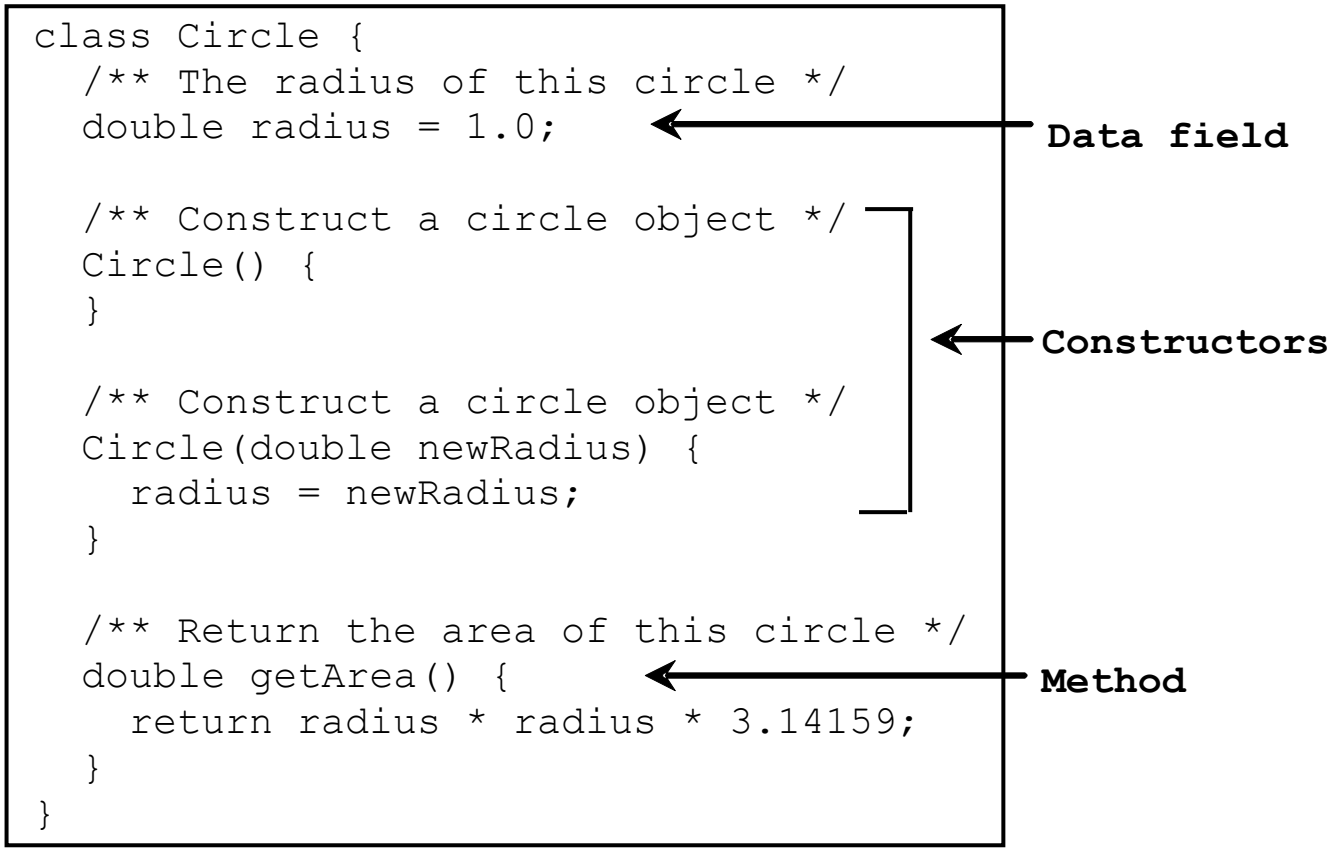
Department of Computer & Information Science

CUNY Brooklyn College

Outline

- In last class, we discussed
  - Defining object
  - Defining class
  - UML class diagram
  - Constructors
- More about constructor
  - The default constructor
  - Overriding the default constructor
  - Overloading constructors
- Garbage collection
- A few classes in the Java Library (Java API)

A Circle Class



Constructors

- A special kind of methods that are invoked when objects are constructed, typically, to initialize the data fields of the objects

```
Circle() {
}

Circle(double newRadius) {
  radius = newRadius;
}
```

Defining Constructors

- Name: constructors must have the same name as the class itself.
- Method parameter: A constructor may or may not have a parameter
  - (parametrized vs. no-arg constructors) A constructor with no parameters is referred to as a *no-arg* constructor.
- Return type: constructors do not have a return type, not even void.
- Invoke constructors: constructors are invoked using the new operator when an object is created.
- Purpose of constructors: constructors play the role of initializing objects.

Creating Objects using Constructors

- Use the new operator
- Examples
  - new Circle()
  - new Circle(25.0)

Default Constructor

- One may write a class without defining a constructor.
  - In this case, Java compiler will provide a no-arg (no argument) constructor with an empty body, called the default constructor
- However, if one provides a constructor with parameters, Java compiler will not create the default constructor
  - But what if you still want to do, e.g., new Circle()?

For each statement, true or false?

1. A no-arg constructor is the default constructor.
2. The default constructor is a no-arg constructor.

Questions

- Constructors
  - Writing constructors
    - Parameterized vs. no-arg constructors (constructors with or without arguments)
  - Using constructors
- Default constructors

Object Referencing Variables

- One may *access objects* via object reference variables (or reference variables)
- 4 steps
    1. Declare reference variable
    2. Create an object
    3. Assign reference to an object to the reference variable
    4. Use the reference variable to access the object

Declaring Object Reference Variables

- To declare a reference variable, use the syntax:
  - ClassName objectRefVar;
- Example
  - Circle c1;

Assigning Reference Variables

- Example

  Circle c1;

  c1 = new Circle();

Declaring/Creating Objects in a Single Step

- Syntax
  - ClassName objectRefVar = new ClassName();
- Example
  - Circle c1 = new Circle();

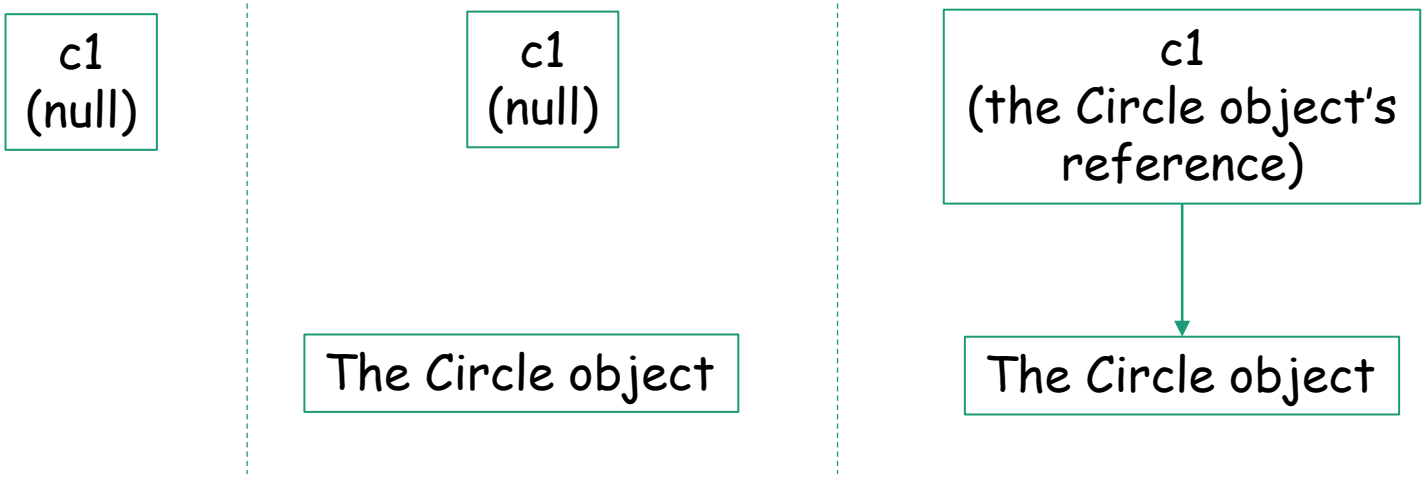
Accessing Objects

- Accessing objects
  - Accessing data fields
  - Accessing methods
- Data fields
  - Syntax: objectRefVar.data
  - Example: c1.radius
- Methods
  - Syntax: objectRefVar.method()
  - Example: c1.getArea()

3 Steps

- Example: one statement, 3 steps
  - Circle c1 = new Circle()

step 1 step 2 step 3



Reference Data Fields

- Data fields of a class can be of reference types
- Example

```
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'
}
```

The null Value

- If a data field of a reference type does not reference any object, the data field holds a special literal value, null.

Default Value for Variables

- Java assign default values to data fields
  - null for a reference type
  - 0 for a numeric type
  - false for a boolean type,
  - '\u0000' for a char type.
- However, Java assigns no default value to a local variable inside a method.

Example: Examining Default Values of Data Fields

- Running the program given the Student class

```
public class TestStudent {
  public static void main(String[] args) {
    Student student = new Student();
    System.out.println("name? " + student.name);
    System.out.println("age? " + student.age);
    System.out.println("isScienceMajor? " + student.isScienceMajor);
    System.out.println("gender? " + student.gender);
  }
}
```

Example: Examining Default Values of Local Variables

- Java assigns no default value to a local variable inside a method.
- When attempting to compile the program, what would you observe?

```
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);
  }
}
```

Primitive Types and Reference Types

- Java has two categories of data types
  - Primitive type
  - Object reference type/reference type

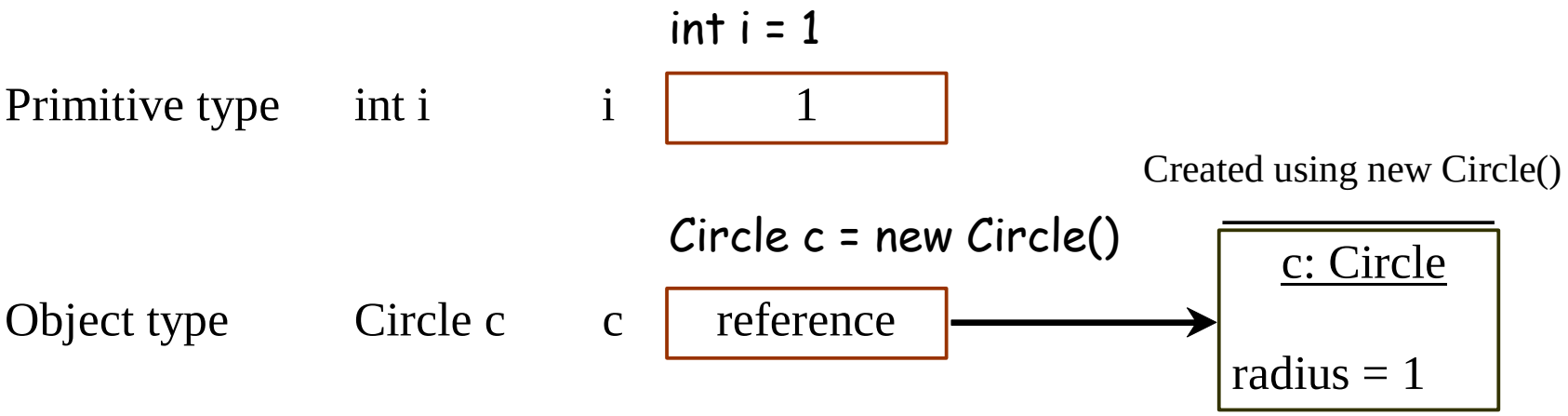
Java Primitive Data Types

- 8 primitive data types

Type	Description	Default	Size	Example Literals
boolean	True or false	False	1 bit	true, false
byte	integer	0	8 bits	(none)
char	Unicode character	\u0000	16 bits	‘a’, ‘u0041’, ‘\101’
short	Integer	0	16 bits	(none)
int	Integer	0	32 bits	-9, -8, 0, 1 2
long	Integer	0	64 bits	3L, 1L, -1L, -3L
float	Floating point	0.0	32 bits	3.14e10f, -1.23e-100f
double	Floating point	0.0	64 bits	1.1e1d, -3.14e10d

Primitive and Reference Types: Difference

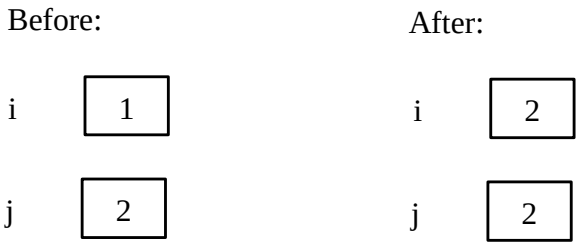
- Illustrate the difference using the example



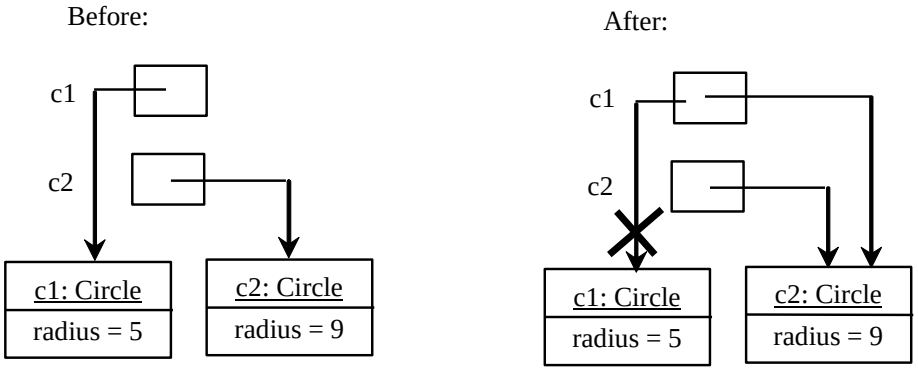
Copying Variables

- Primitive and Reference types

Primitive type assignment i = j

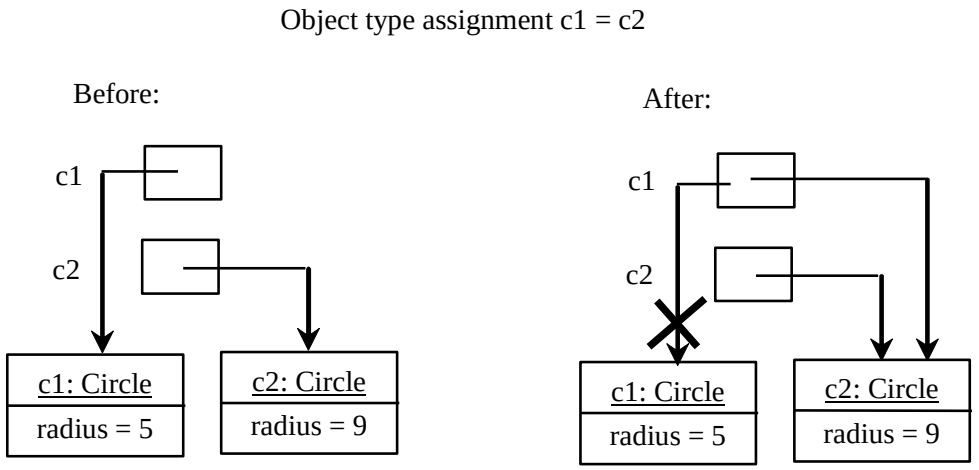


Object type assignment c1 = c2



Garbage Collection

- After c1 = c2, the object previously referenced by c1 is no longer being referenced. This object becomes a garbage. Garbage is automatically collected by JVM.



Garbage Collection: Tip

- If you know that an object is no longer needed, you can explicitly assign null to a reference variable for the object.
- The JVM will automatically collect the space if the object is not referenced by any variable .

Questions

- Primitive and object/(object) reference types
- Accessing objects via reference variables
- Difference between primitive and reference types
- Garbage and garbage collection

In-Class Exercise 1

- Complete exercise 9.5.5 in the textbook (your exercise number may be different)
- Revise the 4 programs so that each of the 4 programs can compile and run
  - Name the 4 classes as ShowErrorsA, ShowErrorsB, ShowErrorsC, and ShowErrorsD instead
- Demo to the instructor when requested.

About CodeLab Exercises

- A tip about CodeLab exercises:

  Unless specified otherwise, when defining a class in CodeLab, add the “public” keyword before the “class” keyword, as in,

  ```
  public class Simple {
  }
  ```

  , add the “private” keyword before data fields, as in,

  ```
  private int hours;
  ```

  , and add the “public” keyword before methods, as in,

  ```
  public int getHigh() { return high; }
  ```