CS202: IT Workshop

Java

OO Concepts: Class, Object

Ref:

1. Harvey Deitel, Paul Deitel, **Java: How to Program**, 9/e, Prentice Hall India.

2. Herb Schildt, **Java: The Complete Reference**, 8/e Tata Mcgraw Hill Education.

Class and Object: Introduction

- We model a real-world scenario and then program the situation (e.g. Library management system)
- Object indicates an entity and Class is a blueprint or a template for that
  - **Book**, **User**, **Faculty**, **Student** can be different classes in LMS
  - ***u1*** is a User → u1 has name, registrationID, age → u1 is an object of class User
  - ***fac1*** is an object of class Faculty → ***fac1*** has noOfPublication
- Class indicates a **type** and object indicates a **variable** (e.g. String str; Faculty fac1;)
- Object is thought of an **instantiation** of a class

Class and Object: Introduction

- Class captures (**encapsulates**) the **features** of the entity and the **operations** on them.

Example:

Features: **name**, **age** for User; **bookTitle** for Book

Operations: **getName()** for User; **issue()** for Book

- **What will become a class for a given problem?**
  - ✓Problem specific; application designer decides that

Signature of a class

Typical look of a Class

```
class classname {
      type instance-variable1;
      type instance-variable2;
      // …
      type instance-variableN;

      type methodname1(parameter-list) {
         // body of method1
      }
      type methodname2(parameter-list) {
        // body of method2
      }
      // …
      type methodnameN(parameter-list) {
        // body of methodN
      }
}
```

Instance variables capture the characteristics

Methods basically operates on instance variables

Methods indicate the operations

Wrapping up of data and the method together in a single unit is called **Encapsulation**

Class and Object in Java

A Java program to demonstrate a Circle class

```
class Circle {

        private int radius;

        public void setRadius(int r) {
                radius = r;
        }
        public double circumference(){
        return 2*3.14*radius;
        }

        public double area(){
        return 3.14*radius*radius;
        }
}
public class CircleDemo {
    public static void main(String args[]) {
        Circle c = new Circle();

        c.setRadius(5);

        System.out.println("Circumference" + c.circumference());
        System.out.println("Area" + c.area());
    }
}
```

radius is an instance variable of class Circle.

its access specifier is **private**; can only be accessed by the members of the same class → **data hiding**

public members of a class can be accessed from outside the class.

An object ***c*** of class Circle is created

Method is called

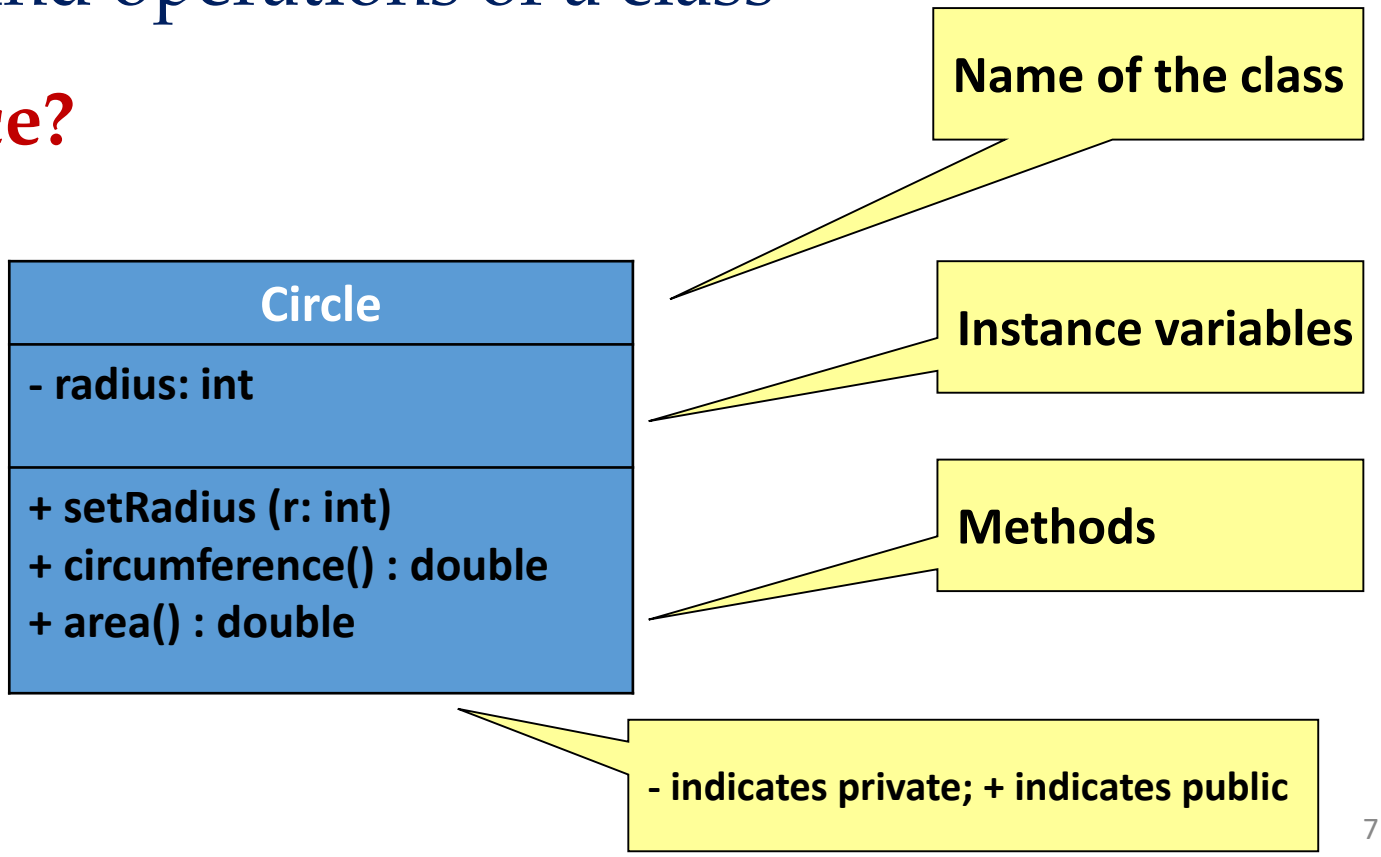
Members are accessed using **<object name>.<membername>**

Access specifiers of members

- **Instance variables** of a class are usually made **private**
- **Methods** are usually made **public**
- **Private** members of a class are accessed by other members of the **same class** only.
  (Class is behaving like a family. C++ supports Friend)
- **Public** members can be accessed from **outside** (other class, method of another class)
- **Protected** members are accessible to the classes of the same **package** and from **subclasses** of a class.

Diagrammatic representation of a Class

- Unified Modeling Language (UML) can be used to model, visualize, document a software system
- UML Class diagram can be used to describe the attributes and operations of a class
- **Importance?**



Constructor in Java

- **A special method**
  - ✓ whose name is same as that of the class
  - ✓ which is automatically called when an object of the class is created
  - ✓ which cannot return any value
- **A class can have multiple constructors**
  - ✓ But their signatures have to be different

e.g. Circle(){ … }, → default constructor
Circle (int r){ … } → parameterized constructor

Circle
- radius: int
<<constructor>> Circle() **<<constructor>> Circle(r: int)** + setRadius (r: int) + circumference() : double + area() : double

Constructor in Java

A Java program to demonstrate a Circle class

```
class Circle {

        private int radius;

        Circle() {
           radius = 0;
        }
        Circle( int r) {
           radius = r;
        }

        public void setRadius(int r) { … }
        public double circumference(){ … }
        public double area(){ … }
}
public class CircleDemo {
        public static void main(String args[ ]) {

           Circle c1 = new Circle();

           Circle c2 = new Circle(5);
           …..
         }
}
```

Default constructor

Parameterized constructor

Default constructor is called

Parameterized constructor is called

OOP Terminologies (We saw in this lecture)

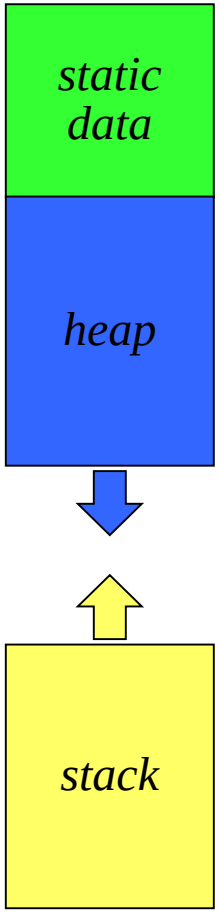
- **Class:** provides a structure, framework
- **Object:** an instantiation of a class
- **Package:** collection of classes
- **Access specifier:** indicates visibility
  - **Private, public, protected**
- **Members of class**
  - **Instance variable**
  - **Method**
- **Constructor:** special method
- **Encapsulation:** Wrapping up data and functions in the same unit

Kinds of variables in Java

- **Instance Variables (Non-Static Fields)**
  - ✓ Each object will have their individual values
- **Class Variables (Static Fields)**
  - ✓ Only one copy exists for a class; all the objects share it.
- **Local Variables**
  - ✓ Variables declared inside a method/function
- **Parameters**
  - ✓ Arguments passed to a method

The Allocation of Memory to Variables

- JVM maintains several memory regions.
- One region of memory is reserved for variables that are never created or destroyed as the program runs, such as **named constants** and other **class variables**. This information is called **static data**.
- Whenever we create a new object, Java allocates space from a pool of memory called the **heap**.
- Each time we call a method, Java allocates a new block of memory called a **stack frame** to hold its local variables. These stack frames come from a region of memory called the **stack**.



Memory allocation for Objects and GC

- When an object is created, memory space is allocated from heap.
- and a reference variable is created which points to the memory location.

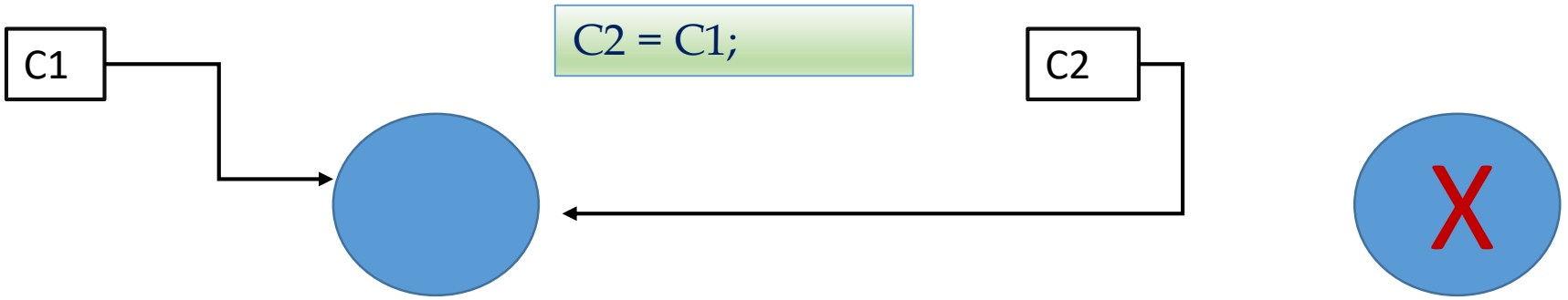
```
Circle C1 = new Circle();

Circle C2 = new Circle();
```

Memory allocation for Objects and GC

- JVM calls automatic memory management module called Garbage collector (GC) when heap becomes almost full.

- Any unreachable object becomes a candidate for GC



Object becomes unreachable

→ Becomes candidate for garbage collection.

Static members in Java

- There are static variables and static methods
- Static variables are typically used for a class-wide information
  - ✓e.g. To count total number of objects created for a class
- Public static members are accessed using class name
  - ✓e.g. Math.random(); Math.PI;
- A static method of a class cannot access its non-static member variable
  - ✓non-static member variable is object specific;

Static members in Java

A Java program to demonstrate static variables

```
class Number {
        int item1;  private static int item2;

        public void fn1() {
        System.out.print( item1 );
        System.out.print( item2 );
        }

        public static void fn2() {
        System.out.print( item1 );
        System.out.print( item2 );
        }

}

public class NumberDemo {
    public static void main(String args[])  {

        Number.fn2();

        Number.fn1();
            }
}
```

item1 is non-static. Thus is cannot be accessed inside a static method **fn2()**

fn1() in not static. Thus it should be accessed using an object.

Did you like the concept of Class and Object?

A. Yes

B. No

C. Somewhat: concept could have been improved