Monsoon 2019

Variables and Methods

Object Oriented Programming

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

Sri City – 517 646, Andhra Pradesh, India

Recap: Objects in JAVA ?

- An entity that has **state** and **behaviour** is known as an object
  - **Examples:** Chair, bike, marker, pen, table, car etc
  - It can be physical or logical
- An object has three characteristics:
  - **State:** represents data (value) of an object
  - **Behaviour:** represents the behaviour (functionality) of an object such as deposit, withdraw and so on
  - **Identity (Internally used):**
    - Signature (unique) of the object
    - Object identity is typically implemented via a unique ID
    - The value of the ID is not visible to the external user
    - But, Internally by JVM to identify each object uniquely

Recap: Naming Conventions

- Just a Guideline (Best Practice)!!

Name	Convention
Class	Should start with uppercase letter and be a noun e.g. String, Color, Button, System, Thread etc
Interface	Should start with uppercase letter and be an adjective e.g. Runnable, Remote
Variables	Should start with lowercase letter e.g. firstName, orderNumber etc
Methods	Should start with lowercase letter and be a verb, for example, actionPerformed(), main(), print(), println() and so on
Constants	Should be in uppercase letter. e.g. RED, YELLOW, MAX_PRIORITY etc
Package	Should be in lowercase letter e.g. java, lang, sql, util etc

Recap: Declaring a Class

```
access specifier class classname {
        type instance-variable1;
        …
        type instance-variableN;

        type methodname1(parameter-list) {
                // body of method 1
        }
        …
        type methodnameN(parameter-list) {
                // body of method N
        }
}
```

Recap: Remember 3 types

Example: Create a Banking Software with functions like

- Deposit
- Withdraw
- Show Balance

Class and interface

```
public class Employee  {
        //code snippet
}

interface Printable  {
        //code snippet
}
```

Package and constant

```
package com.bank;
class Employee  {
        //code snippet
}
class Employee  {
        //constant
        int MIN_AGE = 18;
        //code snippet
}
```

Variables and Methods

```
class Employee {
        //variable
        int id;
        //code snippet
}

class Employee  {
        //method
        void draw()  {
        //code snippet
        }
}
```

Ways to Create Objects?

- **how many ways are there to create an object?**

- There are multiple ways in JAVA
  - Using **new** keyword
  - Using **Class.forName()**
  - Using **clone()**
  - Using Object Deserialization
  - Using **newInstance()** method

Using new keyword

- A class provides a blueprint for objects. So an object is created from a class.

- There are three steps when creating an object:

  - **Declaration** – A variable declaration with a variable name with an object type

  - **Instantiation** – The 'new' keyword is used to create the object

  - **Initialization** – The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Syntax – Using new keyword

- **Using new keyword:**
- The most common way to create an object in java
- Almost 99% of objects are created in this way

- **Syntax:**

```
class_name object_name = new class_name();
```

- **Example**

```
Rectangle obj = new Rectangle();
```

Default Constructor

Two Steps

- Look Closer: Using **new** keyword

  Rectangle obj = **new** Rectangle();

- This above statement combines two steps
- It can be rewritten to show each step clearly as follows:

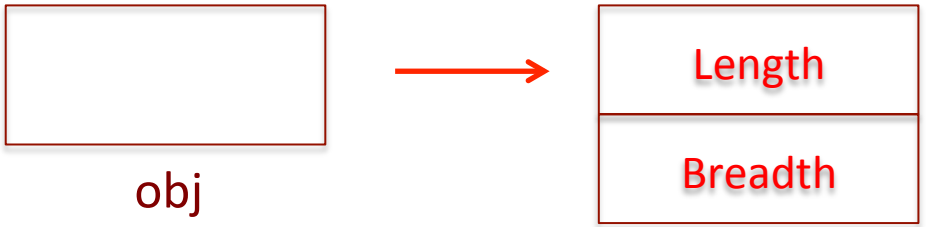
```
Rectangle obj;              // Declaring Reference to Objects
Obj = new Rectangle();   // Allocate a Rectangle Object
```

- The first line declares **obj** as a reference to an object of type **Rectangle**
- At this point, **obj** does not yet refer to an actual object
- The next line **allocates** an object and assigns a reference to it
- Now **obj** acts as a Rectangle object
- But in reality, **obj** simply holds, in essence, the memory address of the actual **Rectangle** object

Illustration

- Using **new** keyword

```
Rectangle obj;
obj = new Rectangle();
```

Statements	Effect
Rectangle obj;	 obj
Rectangle obj;	 obj Rectangle Object

Another Example

- A short customer profile

```
class Customer {
        String name;
        int accountNumber;
        int age;
        double balance;
}
```

- When a reference is made to a particular customer with its property then it becomes an object, physical existence of Customer class

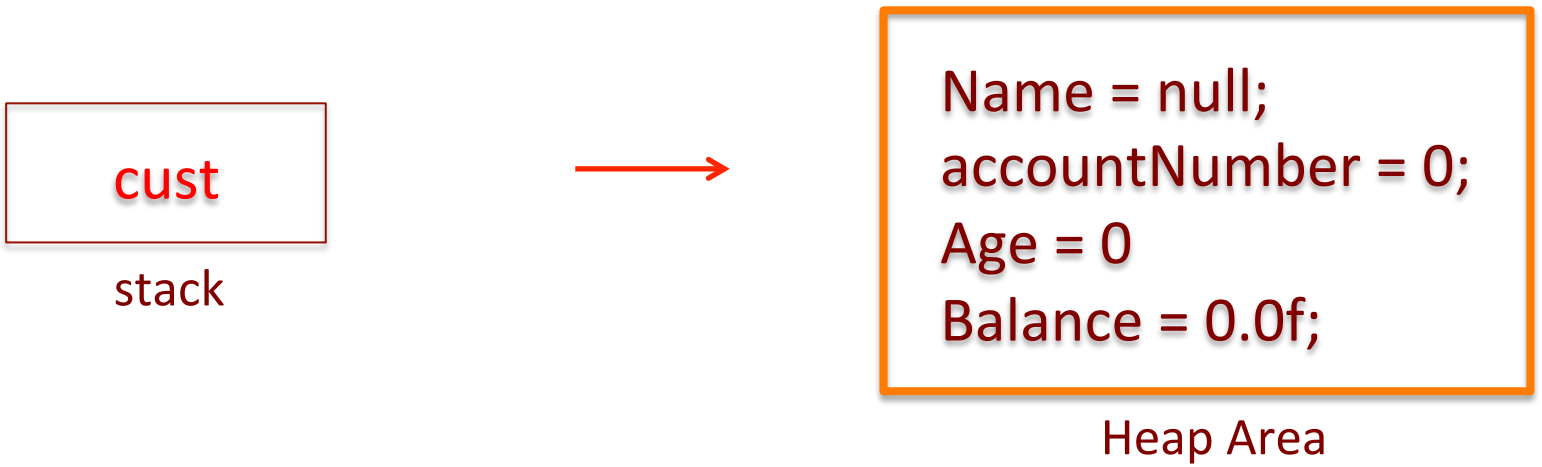
```
Customer cust = new Customer();
```

- cust is instance/object of Customer class
- **new** keyword creates an actual physical copy of the object and assign it to the **cust** variable
- It will have physical existence and get memory in heap area
- The **new** operator **dynamically allocates** memory for an object class Customer

Explanation

Customer cust;

Cust = new Customer ();



Points to Remember

Creating an Object (using new keyword)

- **Step 1:** Declaring reference variable of type class (Rectangle)

  Rectangle obj;

- **Step 2:** Allocation of heap memory by invoking default constructor using new keyword

  new Rectangle();

- **Step 3:** Assigning newly created heap memory allocation to reference variable

  obj = new Rectangle();

- But in practical scenario, we write these three statements in one line, like below

  Rectangle obj = new Rectangle();

Another Way to Create Objects

- Using **Class.forName():**

- Two points to remember:
  - The name of the class is known
  - The class should have a public default constructor

- Then an object can be created as follows:
- Syntax:

```
Rectangle obj;
obj = (Rectangle) class.forName("object").newInstance();
```

Creating Objects – More ways

✧ **Another way to create a new object**

Using clone():

✧ The clone() can be used to create **a copy of an existing object**

✧ Syntax:

Rectangle object = (Rectangle) obj.clone();

Accessing Instance Variables and Methods

- Each object contains its own set of variables
- we should assign values to these variables
- Remember, all variables must be assigned values before they are used.
- Since if we are outside the class, we can not access the instance variables and methods directly

- To do this we must use concerned object and dot operator as shown in the next slide

Variables and Methods

- Instance variables and methods are accessed via created objects
- To access an instance variable, following is the fully qualified path

- /* First create an object */
  Rectangle ObjectReference = new Rectangle();

- /* Now call a variable as follows */
  ObjectReference.variableName;

- /* Now you can call a class method as follows */
  ObjectReference.MethodName(ParameterList);


Variables and Methods (Contd)

- /* Now call a variable as follows */

  ObjectReference.variableName;

- /* Now you can call a class method as follows */

  ObjectReference.MethodName(ParameterList);

- **ObjectReference** is the name of the object
- **variableName** is the name of the instance variable inside the object
- **MethodName** is the name of the method
- **ParameterList** is a comma separated list "actual values" or (expressions)

Some Examples

- Let us take two examples
  - Example – 1:
    Increment a value
  - Example – 2:
    Anonymous Object
  - Example – 3:
    as

First Example

```
pubic class Increment {
        int myCount = 0;
        void increment ( ) {
                myCount = myCount + 1;
        }
        void print ( ) {
                System.out.println ("count = " + myCount);
        }
        public static void main(String[] args) {
                increment c1 = new Increment ( );
                c1.increment ( );    // c1's myCount is now 1
                c1.increment ( );    // c1's myCount is now 2
                c1.print();
                c1.myCount = 0;    // effectively resets c1 counter
                c1.print();
        }
}
```

First Example

```
pubic class Increment {
        int myCount = 0;
        void increment ( ) {
                myCount = myCount + 1;
        }
        void print ( ) {
                System.out.println ("count = " + myCount);
        }
        public static void main(String[] args) {
                increment c1 = new Increment ( );
                c1.increment ( );    // c1's myCount is now 1
                c1.increment ( );    // c1's myCount is now 2
                c1.print();
                c1.myCount = 0;    // effectively re                r
                c1.print();
        }
}
```

Class Name

Variable

Method – Increment()

print() method

Main Method

Output is:
count = 2
count = 0

Anonymous Object

- Anonymous simply means **nameless**
- An object that have no reference is known as anonymous object
- If you have to use an object only once, anonymous object is a good approach
- The anonymous object is created and dies instantaneously
- But, still with anonymous objects work can be extracted before it dies like calling a method using the anonymous object:

- Anonymous Object Instantiation: **new Test()**

Anonymous Object Example

```
class Compute {

        void fact(int n){
                int fact=1;
                for(int i=1;i<=n;i++){
                        fact=fact*I;
                }
                System.out.println("factorial is "+fact);
        }
        public static void main(String[] args) {
                // calling method with anonymous object
                new Compute().fact(5);
        }
}
```

Output is:
Factorial is 120

Assigning Reference & Variables

- We can assign value of reference variable to another reference variable
- Reference Variable is used to store the address of the variable
- Assigning Reference will not create distinct copies of Objects
- All reference variables are referring to same Object

- **Assigning Object Reference Variables does not**
  - Create Distinct Objects
  - Allocate Memory
  - Create duplicate Copy

Example – Object References

```
class Rectangle {
        double length;
        double breadth;
}
class Test {
        public static void main(String args[]) {
        Rectangle r1 = new Rectangle();
        Rectangle r2 = r1;
        r1.length = 10;
        r2.length = 20;
        System.out.println("Value of R1's Length : " + r1.length);
        System.out.println("Value of R2's Length : " + r2.length);
}
```

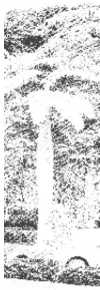
Explanation

```
Rectangle r1 = new Rectangle();
Rectangle r2 = r1;
```

- r1 is reference variable which contain the address of Actual Rectangle Object
- r2 is another reference variable
- r2 is initialized with r1 means – "r1 and r2" both are referring same object , thus it does not create duplicate object , nor does it allocate extra memory
- Thus, any changes made to the object through r2 will affect the object to which r1 is referring, since they are the same object

Copying of Objects

- Two Types:
  - Shallow Copying
  - Deep Copying



Copying of Objects

- **Shallow Copying vs Deep Copying**

  - Copying an object involves getting another object with the same properties of the original.

  - Here, there exists two ways:
    - two objects having their own set of properties (instance variables)

      OR

    - both objects referring the same location of properties.

Shallow Copying

- **Shallow Copying**
  - Shallow copying is the easier of the two styles; here, one object is assigned with another
  - When assigned, both objects refer the same location of variables
  - Both objects refer or share the **same location**
    - If one object changes the value, the other object also gets affected
  - That is, to say straight is no encapsulation exists
  - One small advantage is memory used is less.

Shallow Copying – An Example

```
public class Student {
    int marks;
    public static void main(String args[]) {
        Student std1 = new Student();
        Student std2 = new Student();
        std1.marks = 75;
        std2 = std1; // object to object assignment

        System.out.println("std1 marks before : " + std1.marks);
        System.out.println("std2 marks before: " + std2.marks);
        std1.marks = 85;
        System.out.println("std1 marks after: " + std1.marks);
        System.out.println("std2 marks after: " + std2.marks);
    }
}
```

Deep Copying

- **Deep Copying**
  - The minus of point of shallow copying is over come in deep copying
  - In deep copying, we assign variable by variable of one object to the other
  - Even though, it is tedious, encapsulation exists

Deep Copying – An Example

```
public class Officer {
    int salary;
    public static void main(String args[]) {
        Officer o1 = new Officer();
        Officer o2 = new Officer();
        o1.salary = 5000;
        o2.salary = o1.salary;
        System.out.println("o1 salary before: " + o1.salary);
        System.out.println("o2 salary before: " + o2.salary);
        o1.salary = 6000;
        System.out.println("o1 salary after: " + o1.salary);
        System.out.println("o2 salary after: " + o2.salary);
    }
}
```

Deep Copying - explanation

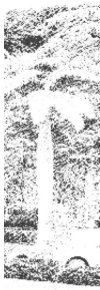
o2.salary = o1.salary;

- In the above statement, salary of o1 is assigned to the salary of o2
- Here, variable to variable is assigned (in the earlier program, object to object is assigned)
- In this case, encapsulation is maintained as both objects have their separate locations for salary variable
- One object does not have any relation (connection) with the other.
- For this reason, as you can observe in the above, when o1 salary is changed to 6000, o2.salary did not get affect
- You can also try with o2 salary and observe o1 salary does not change

Exercises

- **Create Geometric Objects**
  - Perform Basic Operations
  - Apply Transformation
  - Perform getter and setter
  - Extending the Object to Other Shapes

- **Bank Application**
  - Employee
  - Customer
  - ATM
  - Account details
    - Balance enquiry



Assignments / Penalties

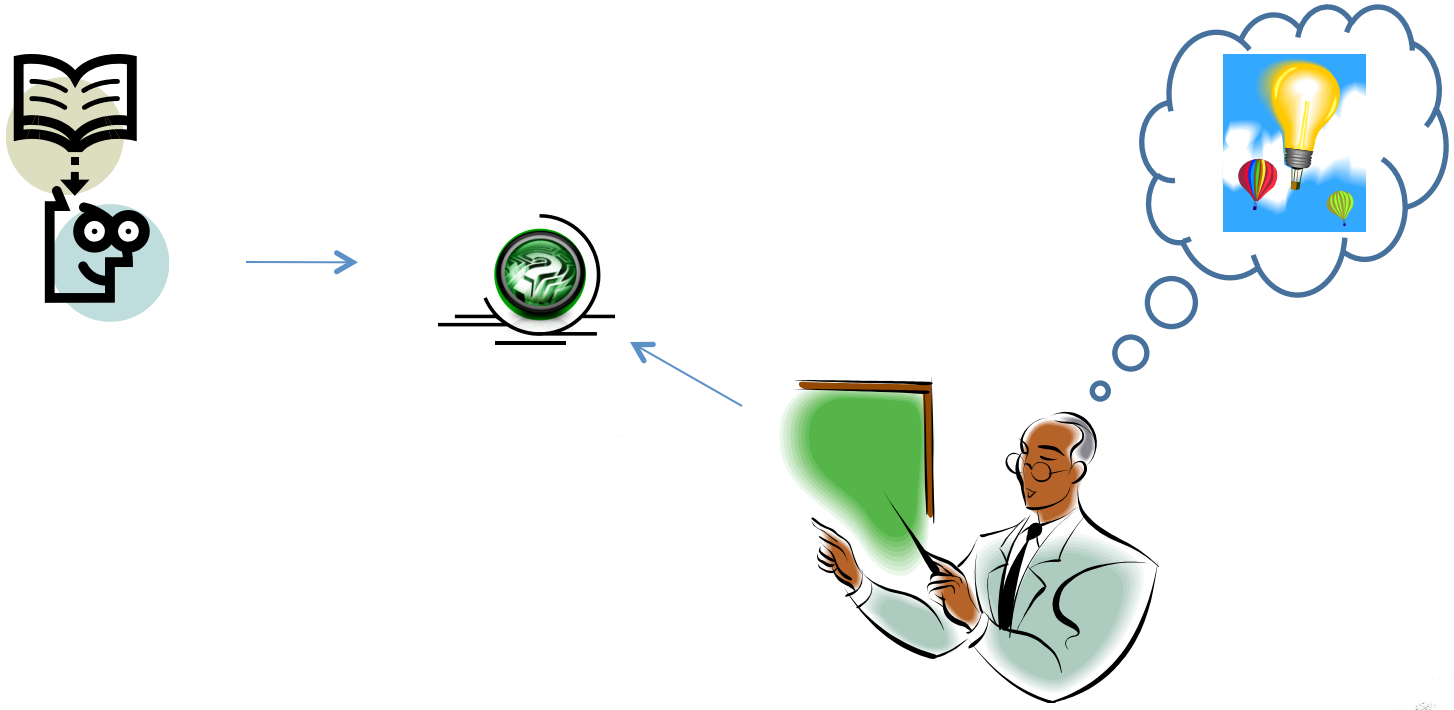
- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties

- Penalties would be heavy for those who involve in:
  - **Copy and Pasting** the code
  - **Plagiarism** (copied from your neighbor or friend – in this case, both will get "0" marks for that specific take home assignments)
  - If the candidate is **unable to explain his own solution**, it would be considered as a "copied case" !!
  - **Any other unfair means** of completing the assignments

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time
  (email is the best way to reach me faster)

Thanks …



… Questions ???