03

CLASS AND OBJECT

OBJECT ORIENTED PROGRAMMING LAB

SESSION - 03

CSE215 | DAFFODIL INTERANTIONAL UNIVERSITY

Object:

- An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical
- Object is any real world thing, that you want to represent in a program with two parts-

1. Properties/state [something that has properties]
2. Actions/behavior [something that does actions]

An object has three characteristics:

- state: represents data (value) of an object.
- behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.
- identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior. Another Object is a Phone which given below,

A phone

What it has?(Properties)

width	width=6.98 or “6.98cm”
height	height=13.6 or “13.6cm”
color	color=“gray”
OS	OS=“Android”
price	price=1000 or “1000$”
brand	brand=“Samsung”
weight	weight=“130g”

What it does?(Actions)

call()	runGames()
sendSms()	enableSharing()
browseInternet()	
runApps()	

Class:

- A class is the representation of common properties and functions of an object.
- It is a template or blueprint from which objects are created.

A class in Java can contain:

- fields
- methods
- constructors
- blocks
- nested class and interface

Following is a sample of a class:

```
public class Dog {
   String breed;
   int age;
   String color;
   void barking() {
   }
   void hungry() {
   }
   void sleeping() {
   }
}
```

A class can contain any of the following variable types.

- Local variables – Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- Instance variables – A variable which is created inside the class but outside the method, is known as instance variable. Instance variable doesn't get memory at compile time. It gets memory at run time when object(instance) is created. That is why, it is known as instance variable. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- Class variables – Class variables are variables declared within a class, outside any method, with the static keyword.

Now look over following example of a class “Person”. It contains data fields and methods.

Where data fields are hairColor, eyeColor, skinColor, shirt, weight and methods are eat(), sleep(), walk(), run(),party().

class Person

hairColor
eyeColor
skinColor
shirt
weight

eat()
sleep()
walk()
run()
party()

Now look over this example.. Here you can see two object. These are “You” and “Your Friend”. See, you and your friend have common property and common method.

You			Your Friend		
					
What do you have?(Properties)		What do you do?(Actions)	What does he/she have?		What does he/she do?
hairColor	="brown"	eat()	hairColor	="darkbrown"	eat()
eyeColor	="black"	sleep()	eyeColor	="black"	sleep()
skinColor	="white"	walk()	skinColor	="tan"	walk()
shirt	="orange"	run()	shirt	="blue"	run()
weight	=70 or "70kg"	party()	weight	=65 or "65kg"	party()

Now just look over what is class and what is object again. Now notice, Class is a list of common properties of some objects. Look the previous example of Class "Person". What you see?? "Person" contains the common properties that you and your friend contains. This is what "Class" is actually. It's a blue print of objects. You and your friend both are objects of "Person" class.

If you aren't clear yet then think of your varsity admission form. Which is same for all but different students fill the data fields with different information. Here "form" is a class and students are objects.

Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

new keyword in Java

The new keyword is used to allocate memory at run time. All objects get memory in Heap memory area.

⌗ How to Create an Object??

There are many ways to create an object in java. They are:

- By new keyword
- By newInstance() method
- By clone() method
- By deserialization
- By factory method etc.

In Java, we create an object by creating an *instance of a class.*

The new operator instantiates a class by allocating memory for a new object of that type. new requires a single argument: a call to a constructor method. Constructor methods are special methods provided by each Java class that are responsible for initializing new objects of that type. The new operator creates the object, the constructor initializes it.

Here's an example of using the new operator to create
a Rectangle object (Rectangle is a class in the java.awt package):

```
new Rectangle(0, 0, 100, 200);
```

In the example, Rectangle(0, 0, 100, 200) is a call to a constructor for the Rectangle class.

The new operator returns a reference to the newly created object. This reference can be assigned to a variable of the appropriate type.

```
Rectangle rect1 = new Rectangle(0, 0, 100, 200);
```

Object and Class Example: main within class

In this example, we have created a Student class that have two data members id and name. We are creating the object of the Student class by new keyword and printing the objects value.

Here, we are creating main() method inside the class.

File: Student.java

```
class Student
{
   int id;  //field or data member or instance variable
   String name;

   public static void main(String args[])
   {
      Student s1=new Student();  //creating an object of Student
      System.out.println(s1.id);  //accessing member through reference variable
      System.out.println(s1.name);
   }
}
```

Here,
Students -> name of the class.
S1 -> object **REFERENCE**
Student() -> Constructor of "Students". [constructor described later]

Object and Class Example: main outside class

In real time development, we create classes and use it from another class. It is a better approach than previous one. Let's see a simple example, where we are having main() method in another class.

We can have multiple classes in different java files or single java file. If you define multiple classes in a single java source file, it is a good idea to save the file name with the class name which has main() method.

```
class Student
{
   int id;
   String name;
}
class TestStudent1
{
   public static void main(String args[])
   {
      Student s1=new Student();
      System.out.println(s1.id);
      System.out.println(s1.name);
   }
}
```

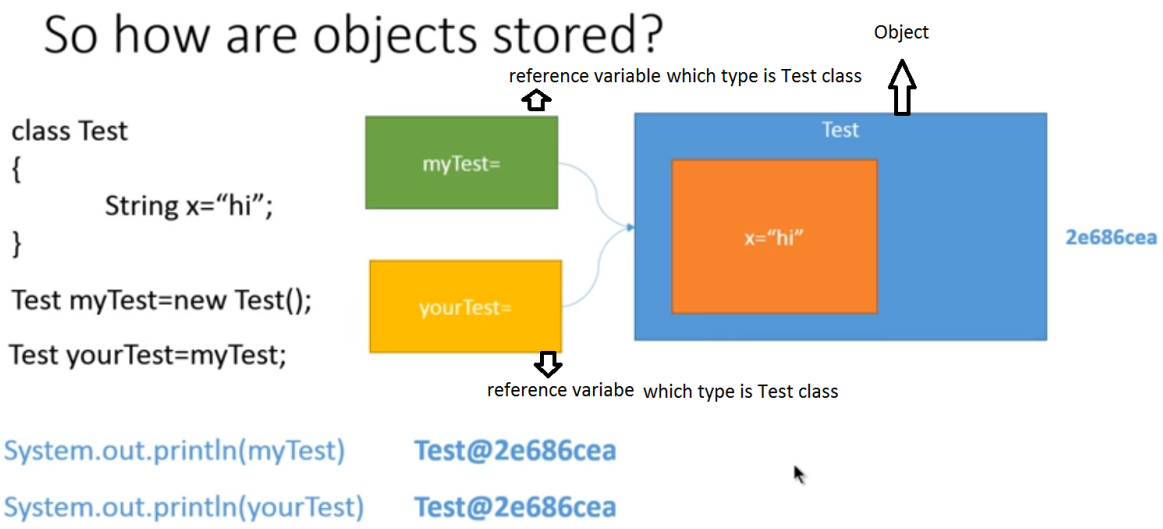
**Look the example below:(see how object created and how reference variable works)

```
Class Test
{
    String x = "hi";
}
Test myTest = new Test();
```

Here we creating an object.

Test = name of the class
MyTest= object REFERENCE
Test() = constructor of Test

- In java, a variable whose type is a class does not actually hold an object.
- It holds the memory location of an object.
- The object itself is stored elsewhere
- Objects can be large. Hence more efficient to store only the memory location of the entire object.



Initialize Object:

There are 3 ways to initialize object in java.

1. By reference variable
2. By method
3. By constructor

1) Object and Class Example: Initialization through reference

```
class Student
{
   int id;
   String name;
}
class TestStudent3
{
   public static void main(String args[])
   {
      //Creating objects
      Student s1=new Student();
      Student s2=new Student();
      //Initializing objects
      s1.id=101;
      s1.name="Sonoo";
      s2.id=102;
      s2.name="Amit";
      //Printing data
      System.out.println(s1.id+" "+s1.name);
      System.out.println(s2.id+" "+s2.name);
   }
}
```

2. Object and Class Example: Initialization through method

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

```
class Student
{
   int rollno;
   String name;
   void insertRecord(int r, String n)
   {
      rollno=r;
      name=n;
   }
   void displayInformation()
   {
      System.out.println(rollno+" "+name);
   }
}
class TestStudent4
{
   public static void main(String args[])
   {
       //Creating objects
      Student s1=new Student();
      Student s2=new Student();
      //Initializing objects by invoking insertRecord()
      s1.insertRecord(111,"Karan");
      s2.insertRecord(222,"Aryan");
      //Printing data
```

```
        s1.displayInformation();
        s2.displayInformation();
    }
}
```

3. Object and Class Example: Initialization through constructor:

```
class Employee{
    int id;
    String name;
    float salary;
    void insert(int i, String n, float s){
        id=i;
        name=n;
        salary=s;
    }
    void display(){
        System.out.println(id+" "+name+" "+salary);
    }
}
public class TestEmployee{
    public static void main(String[] args){
        Employee e1=new Employee();
        Employee e2=new Employee();
        Employee e3=new Employee();
        e1.insert(101,"ajeet",45000);
        e2.insert(102,"irfan",25000);
        e3.insert(103,"nakul",55000);
        e1.display();
        e2.display();
        e3.display();
```

```
    }
}
}
```

Java Constructor:

- Constructor in java is a *special type of method* that is used to initialize the object whose name is same as the class.
- When we create an object of a class using the new operator then java constructor is invoked.
- It is used to perform initializations of instance variables of a class.
- If we don't define a constructor, Java creates a default constructor that can assign default values for all variables in the class. But if you defined your own constructor then Java doesn't provide default constructor.
- When we create an object, its constructor is called before any methods.

Attributes-

- No return type
- Always public
- Can have parameters
- Can be multiple

There are **two** types of constructors:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

Default Constructor:

- Default constructor provides default values to the object like 0,null etc. depending on the type.[int->0, double->0.0, String->null, Boolean-> false)

```
class Student{
```

```
    int id;
    String name;
// no defined constructor, that means Java created default constructor
    void display(){
        System.out.println(id+" "+name);
    }

    public static void main(String args[]){
        Student3 s1=new Student3();
        Student3 s2=new Student3();
        s1.display();
        s2.display();
    }
}
```

Output:

```
0 null
0 null
```

Java parameterized constructor:

- Parameterized constructor is used to provide different values to the distinct objects.

```
class Student{
    int id;
    String name;

    Student4(int i,String n){
```

```
        id = i;
        name = n;
    }

    void display() {
        System.out.println(id+" "+name);
    }

    public static void main(String args[]){
        Student4 s1 = new Student4(161-15-0000,"Tom");
        Student4 s2 = new Student4(161-15-0001,"Jerry");
        s1.display();
        s2.display();
    }
}
```

Output:

```
161-15-0000 Tom
161-15-0001 Jerry
```

Constructor Overloading/ Multiple Constructor:

```
class Student{
    int id;
    String name;

    Student(int i,String n){   //s1 invoked this constructor
        id = i;
```

```
        name = n;
    }
    Student(int i,String n,int a){  //s2 invoked this constructor
    id = i;
    name = n;
    age=a;
    }

    void display() {
        System.out.println(id+" "+name+" "+age);
    }

    public static void main(String args[]){
        Student4 s1 = new Student4(161-15-0000,"Tom");
        Student4 s2 = new Student4(161-15-0001,"Jerry",20);
        s1.display();
        s2.display();
    }
}
```

Output:

```
161-15-0000 Tom 0
161-15-0001 Jerry 25
```

Examples:

Constructor Code:

Rectangle
- height: int - width: int
+ Rectangle () + Rectangle (int, int) + add (Rectangle, Rectangle): Rectangle + display (): void

```
public class Rectangle {
	private int height;
	private int width;
	public Rectangle() {
		this(0, 0); // calling constructor with parameter
	}
	public Rectangle(int height, int width) {
		this.height = height;
		this.width = width;
		// "this" = instance of the same
		// "height" = parameter
		// ".height" = data member
	}
	public Rectangle add(Rectangle a, Rectangle b) {
		// (Rectangle a, Rectangle b) = passing object as a parameter
		// Rectangle = return object

		Rectangle c = new Rectangle(a.height + b.height, a.width + b.width);
		// new=operator to create object
		// Rectangle = calling parameter constructor
```

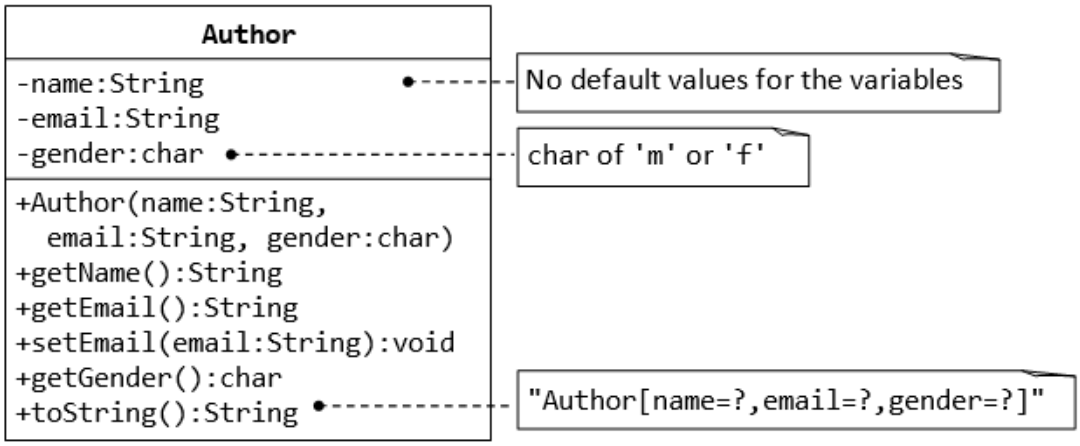
```
            // a.height+b.height,a.width+b.width = added height and
width

            return c;
      }
      public void display() {
            System.out.println("Height is " + height + " and width is " +
width + ".");
      }
}
public class Test {
      public static void main(String[] args) {
            Rectangle a = new Rectangle(10, 15);
            Rectangle b = new Rectangle(3, 5);
            Rectangle c;
            c = a.add(a, b);
            a.display();
            b.display();
            c.display();
      }
}
```

Output is-

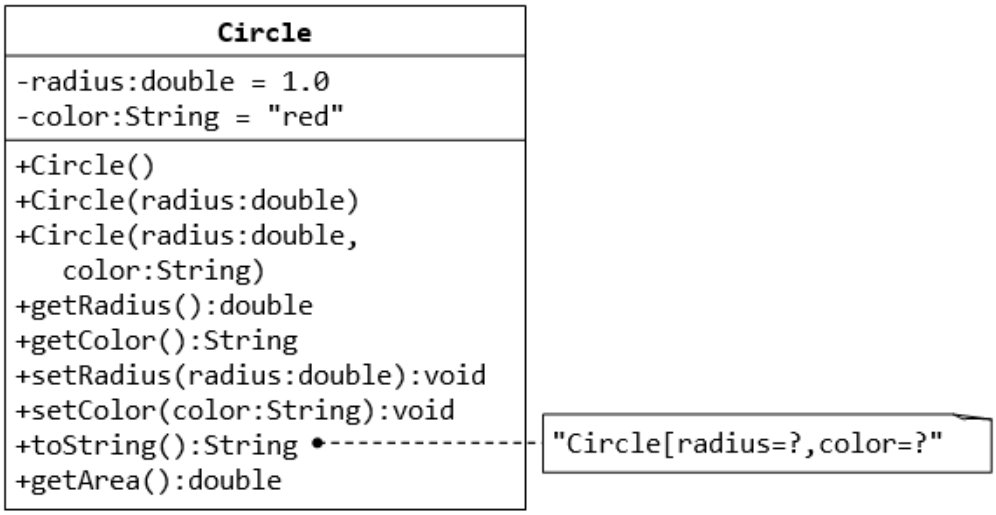
```
Height is 10 and width is 15.
Height is 3 and width is 5.
Height is 13 and width is 20.
```

a. 3.1, 3.2, and 3.3.
b. Draw UMLs from code: Exercise: 3.4, 3.5.

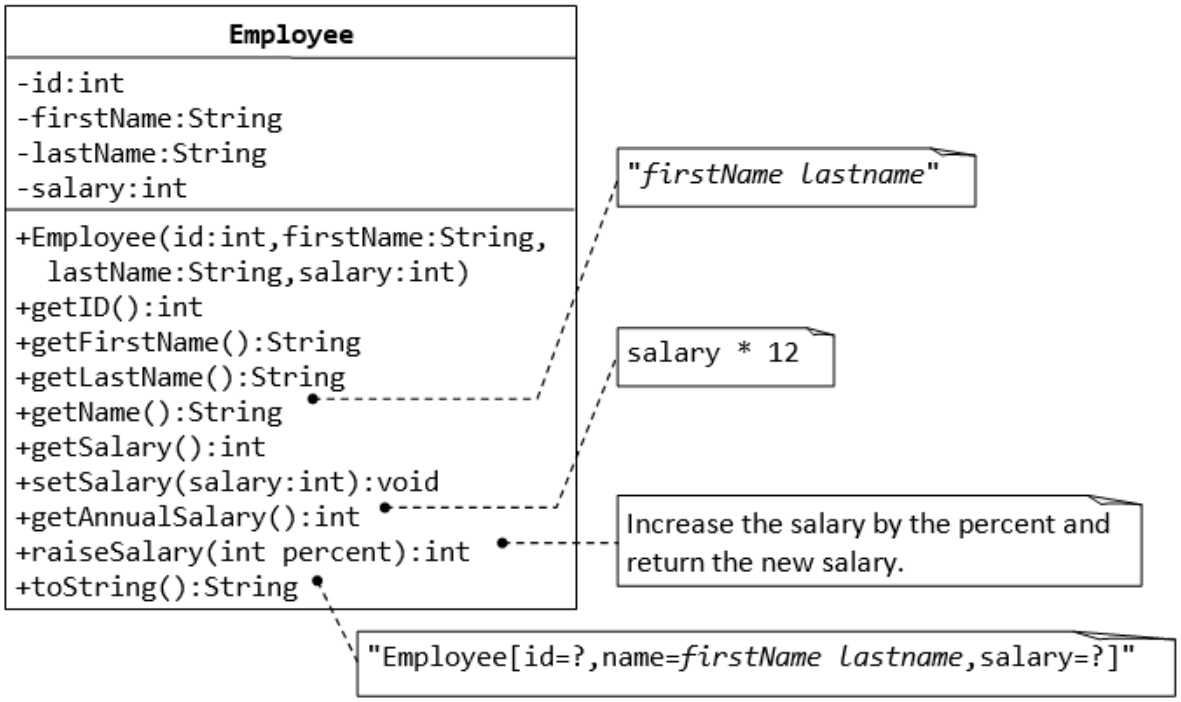
EXERCISE: 3.1:



EXERCISE: 3.2:



EXERCISE: 3.3:



EXERCISE: 3.4:

Design a class **Account** that represents a bank account. Include the following members:

Data members

- Account holders name
- Account number
- Type of account
- Balance amount in the account

Constructor

- To assign initial values(i.e., account holders name, account no, account type and initial deposit)

Methods

- To deposit an amount
- To withdraw an amount after checking balance
- To display the name and balance

EXERCISE: 3.5:

Write a program in Java that uses a class **Student**. Include the following members:

Data members

- Student identification number
- Student name
- Department name
- Course Number
- Marks obtained

Constructor

- Initialize all the variables

Methods

- Take input for all the variables
- Find the grade of the student according to the grading rule of your University
- To display all the variables