UNITIII

Basicsofobjectsandclassesinjava

Java classes is a blueprint or template used to create object.

It serves as a fundamental building block in Java programming, encapsulating data (fields) and behaviors (methods) into a single unit. you specify the attributes and behaviors that objects of that class will possess.
The attributes, also known as fields or instance variables, represent the state or characteristics of objects.
The behaviors, represented by methods, define the actions that objects can perform.

ComponentsofJavaClass

In Java, a class serves as a blueprint for creating objects. It encapsulatesdataandbehaviorintoasingle unit.Herearethe main components of a Java class:

- **Class Declaration**: The class declaration defines the name of the class and any inheritance or interfaces it implements.

```
publicclassMyClass{
   //classbody
```

}

- **Fields (Instance Variables): Fields represent the state or attributes of objects created from the class.**

privateintage;

- **Methods:Methodsdefinethebehaviororactionsthat objects of the class can perform.**

```
public void display() {
   System.out.println("Age:"+age);
}
```

Rulesforcreatinga class

Inordertocreateaclass,theserulesmustbefollowed-

- The“class”keywordshouldbeused.
- The name of the class should start with an uppercase letter.
- The Java file can contain any number of classes but should not have more than one public class. The file name should be named after the public class followed by the “.java” extension.
- Oneclassshouldonlyinheritanothersingleclass.

JavaObjects

A Java object is an instance of a class. It represents a specific realizationof the classblueprint, with itsown unique set of data values for the fields defined in the class.

Objects arecreated usingthe new keywordfollowed by the class name, along with any required arguments to initialize the object's state. Each object created from a class has its own separate memory space allocated for its fields, allowing it to maintain its own state independent of other objects created from the same class.

Objects encapsulate both data (fields) and behavior(methods) into a single unit. They can interact with eachother by invoking methods and accessing fields. In essence, objects are the building blocks of object-oriented programming in Java, allowing developers to model real- world entities and create modular, reusable, and maintainable software components.

Anobjectconsistsof:

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by the methods of an object. It also reflects the response of an object with other objects.
3. **Identity**:Itgivesa unique name to anobjectand enables one object to interact with other objects.

Syntaxofanobject

The syntax for creating an object in Java is:

```
ClassNameobjectName=newClassName();
```

Forexample:

```
CarmyCar=newCar();
```

Here, **Car** is the class name, **myCar** is the object name, and **new Car()** instantiates a new object of the **Car** class.

DifferenceBetweenJavaClassesandObject

There are many differences between object and class. A list of differences between object and class are given below:

No.	Object	Class
1)	Object is an **instance** of a class.	Class is a **blueprint or template** from which objects are created.
2)	Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a **group of similar objects.**
3)	Object is a **physical** entity.	Class is a **logical** entity.
4)	Object is created through **new keyword** mainly e.g. Student s1=new Student();	Class is declared using **class keyword** e.g. class Student{}
5)	Object is created **many times** as per requirement.	Class is declared **once.**
6)	Object **allocates memory when it is created.**	Class **doesn't allocated memory when it is created.**
7)	There are **many ways to create object** in java such as new keyword, newInstance() method, clone() method, factory method and deserialization.	There is only **one way to define class** in java using class keyword.

3Waystoinitializeobject

Thereare3waystoinitializeobjectinJava.

1. Byreferencevariable
2. Bymethod
3. Byconstructor

1) Objectand Class Example:Initialization through reference

Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.

```
classStudent{
intid;
Stringname;
}
classTestStudent2{
publicstaticvoidmain(Stringargs[])

{Students1=newStudent();

s1.id=101;

s1.name="Sonoo";

System.out.println(s1.id+" "+s1.name);//printing members with a white space
}
```

```
}
```

Output:

```
101Sonoo
```

Wecanalsocreatemultipleobjectsandstore informationinitthroughreference variable.

File:TestStudent3.java

```
classStudent{
intid;
Stringname;
}
classTestStudent3{
publicstaticvoidmain(Stringargs[]){
//Creatingobjects
Students1=newStudent();

Students2=newStudent();
//Initializing
objectss1.id=101;
s1.name="Sonoo";
s2.id=102;
s2.name="Amit";
```

```
//PrintingdataSystem.out.println(s1.id+""+s1.name);

System.out.println(s2.id+" "+s2.name);

}
}
```

Output:

```
101 Sonoo
102 Amit
```

1) 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 thedisplayInformation() method.

File:TestStudent4.java

```
class

Student{int

rollno;

Stringname;

voidinsertRecord(intr,Stringn)

{

rollno=r;
```

```
name=n;
}
void
displayInformation(){System.out.println(rollno+""+name);}
}
classTestStudent4{
public static void
main(String args[]){Student
s1=new Student(); Student
s2=new Student();
s1.insertRecord(111,"Karan"
);
s2.insertRecord(222,"Aryan"
);s1.displayInformation();
s2.displayInformation();
}
}
```

Output:

2) Object and Class Example: Initialization through a constructor

WewilllearnaboutconstructorsinJavalater.

ObjectandClassExample:Employee

Let'sseeanexamplewherewearemaintainingrecordsofemployees.

File:TestEmployee.java

```
classEmployee{
intid;
Stringname;
floatsalary;
voidinsert(inti,Stringn,floats)

{id=i;

name=n;

salary=s;

}
void display(){System.out.println(id+" "+name+"
"+salary);}
}
publicclassTestEmployee{
public static void

main(String[]args){

Employee e1=new

Employee(); Employee

e2=new Employee();
```

```
Employeee3=newEmployee();
e1.insert(101,"ajeet",45000);
e2.insert(102,"irfan",25000);
e3.insert(103,"nakul",55000);
e1.display();
e2.display();
e3.display();
}
}
```

Output:

ObjectandClassExample:Rectangle

Thereisgivenanotherexamplethatmaintainsthe recordsofRectangleclass.

File:TestRectangle1.java

```
classRectangle{
intlength;
intwidth;
voidinsert(intl,
int w){length=l;
width=w;
}
```

```
voidcalculateArea(){System.out.println(length*width);}
}
classTestRectangle1{
publicstaticvoidmain(Stringargs[])
{Rectangle r1=new Rectangle();
Rectangler2=new
Rectangle();
r1.insert(11,5);
r2.insert(3,15);
r1.calculateArea();
r2.calculateArea();
}
}
```

Output:

```
55
45
```