Fundamentals

## Class

A class is a blueprint or prototype from which objects are created. It contains properties and method.

### Declaring classes:

[access modifiers] class MyClass [extends MySuperClass] [implements interface1, interface2..]{

//fields, Constructor and method declarations

//instance and static blocks

//Inner classes

}

### Key Points:

In the above syntax, the content which is inside the square braces is optional.

Modifiers such as public, private etc…

Class Name must be started with capital letter only

Class body must be surrounded within the curly braces.

A class can only extend one parent.

A class can implement more than one interface.

## Object

An object is a real world entity which contains below two characters.

1. State
2. Behaviour

### Creating Objects

In general, we will be using the new operator to create an instance of a class.

Example: Student s1 = new Student(1,”Vivan”);

### Key Points:

Above statement is divided into three sections.

1. Declaration

Syntax: type name;

Example: Student s1;

1. Instantiation

**new** operator

1. Intialization

The new operator requires a single, postfix argument: a call to a constructor

Example: Student(1,”Vivan”)

## Defining Methods

Syntax:

Modifiers return-type methodName(Parameters list) throws ExceptionsList {

}

Example: Public double add(int a, float b){

}

### Method Signature

The combination method name and parameter type list is known as method signature.

Example: add(int a, float b)

## OOPS concepts

## Abstraction

Hiding internal implementation and showing only the set of services which are required is called as abstraction. In java, to achieve this we are going to use abstract class and intefaces.

### Abstract Class

An abstract class is a class which is incomplete and it may contain zero or more abstract methods.

If a class contains abstract method then that class must be declared as abstract.

Example 1: public abstract class AbstractClassDemo {

public abstract void display();

}

Example 2: public abstract class AbstractClassDemo {

public void display();

}

#### Key Points:

We can’t create an instance for the abstract class. If we are trying to create then we will get CTE.

We can have constructors in the abstract class.

An abstract class can be extended by sub classes.

AbstractClassSample.java

In the above program, we defined an abstract class called Laptop with one abstract method in it.

### Interface

An interface is a group of related methods with empty bodies. These methods are nothing but the services available. We are using **implements** key word to implement an interface.

#### Key Points:

All interface methods are by default public and abstract.

All interface variables are by default public static final.

We will use implements key word to implement an interface.

A class can implement any number of interfaces.

An interface can be implemented by any number of classes.

To declare an interface we are using the **interface** key word.

While implementing interface method it must be declared as public because weakering of the acess modifier not allowed.

If a class is not implementing all the methods declared inside the interface, then we must declare that class as abstract class.

Example 1:

interface MyInterface {

void display();

}

public class InterfaceSample implements MyInterface {

**private** void display(){

statement(s)

}

}

Example 2:

interface MyInterface {

void display1();

void display2();

}

public **abstract** class InterfaceSample implements MyInterface {

private void display1(){

statement(s)

}

}

InterfaceSample.java

The above code will illustrate how to create and implement an interface.

InterfaceSample2.java

In the above program, our class implemented two interfaces and also gives an idea on how to access the interface variables.

## Encapsulation

Encapsulation is the process of binding of data (i.e. methods and variables) into a single unit called as class. It is achieved in java by using private, protected an public key words.

## Polymorphism

Polymorphism as per the dictionary definition one thing having many different forms/stages.

Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class.

### Overloading

Methods with in a class can have same name but they must contain different parameter list.

OverloadingSample.java

In the above program, display() methods are overloaded methods. Bcoz it contains same method name but different argument list.

### Overriding

In the subclass, if an **instance** method has the same declaration like in the parent class then we call that method as overridden method.

Key Points:

Private method seems to be overridden but it’s not.

We can’t override the final method.

**Co variant return types:**

An overriding method can also return a subtype of the type returned by the overridden method.

This concept is only applicable for objects but not for primitives.

**Weakering of access modifier:**

Here, we should not decrease the size of the access modifiers.

**Method Hiding:**

In the subclass, if an **static** method has the same declaration like in the parent class **static** method then we call that as method hiding.

Example:

Object getParentInfo() --- > String getParentInfo()

Long getParentInfo() --- > int getParentInfo() // co varaiant return type not applicable - primitives

OverrridingSample.java

In the above program, we covered all the mentioned scenarios.

## Inheritance

In java, classes can be inherited from another class, so that it can get all the fields and methods from the parent class.

**Sub Class:** A class that is derived from another class is called a subclass (also a derived class, extended class, or child class).

**Super Class:** The class from which the subclass is derived is called a superclass (also a base class or a parent class)

Inheritance represents a **IS-A** relationship.

### Key Points:

Java.lang.Object class acts as parent/root class for all the classes.

A subclass inherits all of the public and protected members of its parent class except **private.**

You can declare a field in the subclass with the same name as the one in the superclass, thus hiding it (not recommended).

You can declare new methods in the subclass that are not in the superclass.

We are using the **extends** keyword to get the properties of parent class.

Multiple inheritance is not supported by java.

InheritanceSample1.java

In this example, we created an Employee class which act as parent class and OracelEmployee class extends Employee class.

### Diamond Problem

As we already know, java doesn’t support multiple inheritance to avoid the method ambiguity.

DiamondProblemSample\_V.java

In the above sample, we are not sure from which class we are getting the eat() method.

## Variables:

An object stores its state in fields. In java, the terms **field** and **variable** are interchangeable.

Example: int id = 26;

Here,

int is known as data type.

id is known as variable name.

26 is known as literal i.e. value that we are used to assign to a primitive type.

### Variable Naming Convention:

Variable names are case-sensitive.

It should start with a letter, the dollar sign "$", or the underscore character "\_".

We don’t have any length limit.

### Types of Variables:

#### Instance Variables (Non-Static Fields)

Objects store their individual states in instance variables. The values of instance variables are unique to each instance/object of a class.

#### Class Variables (Static Fields)

We are using the static key word to define the class variables. This will create only one copy of variable even though we create N number of objects for the class.

#### Local Variables

Method will store their temporary state in the local variables. Local varaibles are visible with in that method only.

#### Parameters

Parameters are nothing but the arguments to any method or constructor. In the below, **args** is an parameter to main method.

Example: public static void main(String[] args).

## DataTypes

We know that java is **strictly typed** programming language. i.e We must declare all the variables before we are going to use them.

**byte**: The byte data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive).

**short**: The short data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive).

**int**: By default, the int data type is a 32-bit signed two's complement integer, which has a minimum value of -231 and a maximum value of 231-1.

**long**: The long data type is a 64-bit two's complement integer. The signed long has a minimum value of -263 and a maximum value of 263-1.

**float**: The float data type is a single-precision 32-bit IEEE 754 floating point.

**double**: The double data type is a double-precision 64-bit IEEE 754 floating point.

**boolean**: The boolean data type has only two possible values: true and false. Use this data type for simple flags that track true/false conditions.

**char**: The char data type is a single 16-bit Unicode character. It has a minimum value of '\u0000' (or 0) and a maximum value of '\uffff' (or 65,535 inclusive).

|  |  |
| --- | --- |
| **Data Type** | **Default Value (for fields)** |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| String (or any object) | Null |
| boolean | False |

## Key Words.

These are the reserved words which contains specific functionality in java.

Key words for data types: boolean, byte,char, short, int , long, float, double.

Key words for access: private, protected and public.

Key words for modifiers: public, private, protected, static, final, strictfp, abstract, synchronized, volatile, native, transient.

Key words for exception handling: try, catch, finally, throw, throws

Key words for flow control: if, else, switch, case, default, break, do, while, for, continue.

Key words for class and object: package, import, class, extends, interface, implements, new, return, instanceof, this, super.

Key words to assigned values: true, false, null.

Unused key words: goto, const.

### Final

This key word is applicable to class, methods and variables.

**Final class:** If a class declared with final key word, then we can’t create a subclass for that class.

**Final method:** If a method declared with final key word, then we can’t override that method in the child class.

**Final variable:** If a variable declared with final, then we should assign the value at the time of declaration itself and we are not allowed to modify that value.

FinalKeywordSample.java

### Static

This key word is used with class, method, variable, block.

Static class: This is known as static nested class. We covered this topic in the Nested classes section.

Static method: Overriding static methods is known as method hiding.

Static variable: It contains **only one copy** of the variable at a class level, even though we created N number of instance of that class.

Static block: This block of code is used to initialize the static fields.

StaticKeyWordSample.java

### Super

This key word is used to access the methods and constructor of the super class from sub class.

SuperKeywordSample.java

The above sample will explain how to access the super class functionality from sub class.

## Arrays

An array is a container which holds either primitive type/object type with fixed size .

Each item in an array is called an element.

Each element is accessed by its numerical index.

Index value starts with **0**.

Example 1:

int[] intArr; // declares an array of integers

intArr = new int[10]; // allocates memory for 10 integers

intArr [0] = 100; // initialize first element

intArr [1] = 200; // initialize second element

### Single Dimensional Array

OneDimensionalArraySample.java

In the above example, we mentioned few ways to create and initialize the array.

### Multi Dimensional Array

In java, multi dimensional arrays are known as array of arrays.

Example: int intArr[ ][ ] = new int[4][5];

MultiDimensionalArraySample.java

In the above example, we are creating a two dimensional array and also covered object array scenario.