**Java JDK -> java 8 and greater version**

[**https://www.oracle.com/in/java/technologies/downloads/#jdk17-mac**](https://www.oracle.com/in/java/technologies/downloads/#jdk17-mac)

**JDK and JRE 🡪 two items the jave will be able to run on your machine**

**Java – Basics**

Java is an Object-Oriented Language. As a language that has the Object Oriented feature, Java supports the following fundamental concepts:

* Polymorphism – Overloading and Overriding
* Inheritance
* Encapsulation
* Abstraction
* Classes
* Objects
* Instance
* Method

Create a project in Eclipse,

Open Eclipse 🡪 File 🡪 New Project 🡪 Project 🡪 Java Project 🡪 src 🡪 create a package 🡪 Create a class

**What are Class and Objects?**

Class is a template/blue print that describes an object.

Object has a state and behaviour.

Object can be created only for a class.

Syntax for Object:

BaseClass nameofObject = new ReferenceClass();

**What are constructors?**

Constructors are special type of method which is used to initialise an object.

Basically it constructs the value i.e. provides data for the object that is why it is known as constructor.

Rules for Constructors:

1. Constructor name must be same as its class name
2. Constructor must have no explicit return type

Default Constructor:

Default constructor provides the default values to the object like 0, null etc. depending on the type.

Parameterised Constructor:

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

|  |  |
| --- | --- |
| **Java Constructor** | **Java Method** |
| Constructor is used to initialize the state of an object. | Method is used to expose behaviour of an object. |
| Constructor must not have return type. | Method must have return type. |
| Constructor is invoked implicitly. | Method is invoked explicitly. |
| The java compiler provides a default constructor if you don't have any constructor. | Method is not provided by compiler in any case. |
| Constructor name must be same as the class name. | Method name may or may not be same as class name. |

**What are Modifiers?**

Modifiers/Access Modifiers are keywords that you add to those definitions to change their meanings.

There are four Types of Access Modifiers,

1. Private - you cannot create the instance of that class from outside the class.
2. Default – It is accessible only within package
3. Protected - It is accessible within package and outside the package but through inheritance only
4. Public- It is accessible everywhere

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only by using inheritance** | **outside package** |
| **Private** | Y | N | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

**Variables:**

**Primitive Data Types:**

|  |  |  |
| --- | --- | --- |
| byte | 1 byte | Stores whole numbers from -128 to 127 |
| short | 2 bytes | Stores whole numbers from -32,768 to 32,767 |
| int | 4 bytes | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits |
| boolean | 1 bit | Stores true or false values |
| char | 2 bytes | Stores a single character/letter or ASCII values |

**Non-primitive data types:**

String, Arrays and Classes

**Data Types:**

* **String 🡪 Words** 
  + **Example : “selenium”**
* **Int 🡪 numbers** 
  + **Example: 1234**
* **Boolean 🡪 true/false** 
  + **Example : true**
* **Float 🡪 decimal values ends with F** 
  + **Example: 23.23f**
* **Double 🡪 Decimal values but memory is shorter than float**
  + **Example: 23.23**
* **Char 🡪 letters** 
  + **Example: ‘q’**
* **Long 🡪 denotes the numbers** 
  + **Example: 1224234324345**
* **Short 🡪 denotes the numbers** 
  + **Example: 1224234324345**

**Byte:**

1. Byte data type is a 8-bit signed two's complement integer
2. Minimum value is : -128 (-2^7)
3. Maximum value is : 127 (inclusive)(2^7 -1)
4. Default value is : 0
5. Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an int

**Methods:**

It is the place where we need to write the logics. Usually methods are user defined.

Syntax:

accessModifiers returntype nameofmethod(){

}

Public void Addition(){

}

Public int Multiply(){

return 0;

}

**What is abstract in java?**

A class that is declared with abstract keyword is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

**OOPS – Java**

**Encapsulation:**

It is the process of wrapping the data (variables) and code acting (method) on the data together in a singe unit.

In encapsulation the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class, therefore it is also known as data hiding.

Example: getter and setter in coding

**Abstraction:**

Abstraction is a process of **hiding the implementation details** from the user, only the functionality will be provided to the user.

It will be declared before the class.

A class which contains the **abstract** keyword in its declaration is known as abstract class.

* Abstract classes may or may not contain abstract methods ie., methods without body ( public void get(); )
* But, if a class have at least one abstract method, then the class must be declared abstract.
* **If a class is declared abstract it cannot be instantiated.**
* To use an abstract class you have to inherit it from another class, provide implementations to the abstract methods in it.(you need to use **extends** keyword)
* If you inherit an abstract class you have to provide implementations to all the abstract methods in it.

**Inheritance:**

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

**Extends** keyword is used to inherit the properties of the class.

**Super keyword** is used to differentiate the members of the superclass from the members of subclass.

**Polymorphism:**

Polymorphism is the capability of a method to do different things based on the object that it is acting upon.

**Method Overloading:**

It is possible to define two or more methods of same name in a class, provided that there argument list or parameters are different. This concept is known as Method Overloading.

1. To call an overloaded method in Java, it is must to use the type and/or number of arguments to determine which version of the overloaded method to actually call.
2. Overloaded methods may have different return types; the return type alone is insufficient to distinguish two versions of a same method.
3. When Java encounters a call to an overloaded method, it simply executes the version of the method whose **parameters match the arguments** used in the call.
4. It allows the user to achieve **compile time polymorphism.**
5. An overloaded method can throw different exceptions.
6. It can have different access modifiers.

**Rules for Method Overloading**

1. Overloading can take place in the same class or in its sub-class.
2. Constructor in Java can be overloaded
3. Overloaded methods must have a different argument list.
4. Overloaded method with same name but different parameters.
5. The parameters may differ in their type or number, or in both.
6. They may have the same or different return types.
7. It is also known as compile time polymorphism.
8. Same method name but different argument list.
9. Different access modifiers are possible

**Method Overriding:**

Child class has the same method as of base class. In such cases child class overrides the parent class method without even touching the source code of the base class. This feature is known as method overriding.

**Rules for Method Overriding:**

1. applies only to inherited methods
2. object type (NOT reference variable type) determines which overridden method will be used at runtime
3. Overriding method can have different return type
4. Overriding method must not have more restrictive access modifier
5. Abstract methods must be overridden
6. Static and final keywords cannot be overridden
7. Constructors cannot be overridden
8. It is also known as **Runtime polymorphism**.
9. Same method name and same parameters.

**Collections in Java:**

**Collections in java** are a framework that provides architecture to store and manipulate the group of objects (similar objects).

All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Java Collection simply means a single unit of objects.

**Package used: import java.util.;**

* **List/ArrayList**
* **List/LinkedList**
* **Set/HashSet**
* **Table/HashTable**
* **Map/HashMap**

**There are two ways to search/loop inside a collection either by using iterator or for each loop or traditional for loop**

**Java ArrayList Class:**

* Java ArrayList class uses a dynamic array for storing the elements.It extends Abstract List class and implements List interface.
* **Java ArrayList class can contain duplicate elements.**
* **Java ArrayList class maintains insertion order.**
* **It accepts null values**
* Java ArrayList class is non synchronized.
  + Not thread safe
* **Java ArrayList allows random access because array works at the index basis.**
* In Java ArrayList class, **manipulation is slow** because a lot of shifting needs to be occurred if any element is removed from the array list.

**Java LinkedList class:**

* Java LinkedList class uses doubly linked list to store the elements. It extends the Abstract List class and implements List and Deque interfaces.
* **Java LinkedList class can contain duplicate elements.**
* **Java LinkedList class maintains insertion order.**
* **It accepts null values**
* Java LinkedList class is non synchronized.
  + It’s not thread safe
* **In Java LinkedList class, manipulation is fast because no shifting needs to be occurred.**
* Java LinkedList class can be used as list, stack or queue.

**Java HashSet class:**

* uses hashtable to store the elements. It extends Abstract **Set** class and implements Set interface.
* **contains unique elements only.**
* **Doesn’t maintain insertion order**
* **It has null values**

**Hashmap:**

* A HashMap contains values based on the key. It implements the **Map** interface and extends Abstract Map class.
* It maps the data as **key and value pair**
* **It contains only unique elements.**
* **It may have one null key and multiple null values.**
* **It maintains no insertion order.**
* Non-synchronised 🡪 not thread safe

**Java Hashtable class:**

* A Hashtable is an array of list. Each list is known as a bucket. The position of bucket is identified by calling the hashcode() method. A Hashtable contains values based on the key. It implements the Map interface and extends Dictionary class.
* **It contains only unique elements**.
* **It may have not have any null key or value.**
* **It doesn’t maintain insertion order**
* It is synchronized 🡪 threadsafe

Example: Hashtable

 public static void main(String args[]){

**Hashtable<Integer,String> hm=new Hashtable<Integer,String>();**

**hm.put(100,"Amit");**

**hm.put(102,"Ravi");**

**hm.put(101,"Vijay");**

**hm.put(103,"Rahul");**

**for(Map.Entry m:hm.entrySet()){**

**System.out.println(m.getKey()+" "+m.getValue());**

  }

}

**Arraylist vs Linkedlist:**

|  |  |
| --- | --- |
| **ArrayList** | **LinkedList** |
| 1) ArrayList internally uses **dynamic array** to store the elements. | LinkedList internally uses **doubly linked list** to store the elements. |
| 2) Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory. |
| 3) ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. |
| 4) ArrayList is **better for storing and accessing** data. | LinkedList is **better for manipulating** data. |

Example:

public static void main(String args[]){

List<String> al=new ArrayList<String>();//creating arraylist

al.add("Ravi");//adding object in arraylist

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

List<String> al2=new LinkedList<String>();//creating linkedlist

al2.add("James");//adding object in linkedlist

al2.add("Serena");

al2.add("Swati");

al2.add("Junaid");

System.out.println("arraylist: "+al);

System.out.println("linkedlist: "+al2);

}

Note:

Difference between abstract and implements:

Purpose of class:

Abtract class-> baseclass and that will be the abstract class and where the method is referenced to/ where the method is implemnented is the reference class

Interface -> multiple inheritance so the base and reference remains the same

IMplementation:

abstract mentos have conceret implementation

interface - it includes the methos isgnatures default and static

Inheritance :

abstract methods needs extends keyword

interfece needs implements keyword

fields level:

abstract cannot have static inthe method with implementation

interface it can have static fields

constructor:

abstract -> can have constructor

interface -> cannot have cosntrcutors

when to use what?

Abstract class:

1.share code among severale related and closely associated class

2. you do not require non-static and non-final fields

3.define base class with some common functionality

Interface:

1.need to represent multiple inheritance

2.when the multiple classes that may not be closely related but need to share the method names

3.define a contract for classes without implementing a particular structure

Generics in Java:

* 1. Way to create a class,interface and methods that can operate on any data type on compile time
  2. Flexible and reusable code
  3. Parameter -> parameter is mandatory represented like <T>
  4. Main purpose is re-usability