1.Static Keyword:

* Static is used for memory management.
* It creates the memory once in static memory.
* It can be used with variable , method & block.

Static Variable:

* Creates memory once and shares the values.

Static Method:

* Cannot be overridden

Static variable & Methods :

* Can be directly accessed by calling its class name or interface name.

Static Block:

* Static block will be executed before the main method execution.

2. Different between Final, Finally, Finalize

|  |  |  |  |
| --- | --- | --- | --- |
| Key | Final | Finally, | Finalize |
| Definition: | Final is a keyword is used apply restriction & methods and variables | Finally, is an exception handling | Finalize is a pre-defined method.  In object class |
| Functionality | Once Variable declared it will be constant and cannot be modified.  Final method cannot be overridden.  Final class cannot be inherited. | Finally, block is used to execute even if the exception occurs or not. | When ever the object is gone into garbage collection.  System.gc(); |
| Execution | Final Method is executed when we call it. | Finally, block is executed as soon as try catch block is executed.  It is an execution does not depend on exception |  |

3. What is string in how many ways u can declare a string?

String is an immutable class Once defined cannot be changed.

sequence of characters within double quotes

It has some pre defined Like :

toUpperCase()

toLowerCase()

* String Literal,

String a=” Santhosh”;

* Using New Keyword

String s = new String(“Welcome”);

4.Why String is immutable?

String mutable Once declared cannot be changed.

**class** GFG {

**public** **static** **void** main (String[] args) {

    String str="Knowledge";

      String s=str;

      str=str.concat(" Base");

      System.out.println(str);

    }

}

When the above statement is executed, the VM takes the value of String str, i.e. “knowledge” and appends ” base”, giving us the value “knowledge base”. Now, since Strings are immutable, the VM can’t assign this value to str, so it creates a new String object, gives it a value “knowledge base”, and gives it reference str.

5.Difference String Buffer & String Builder:

String buffer & String builder is used to convert a immutable string to mutable string

String Buffer:

String Buffer is present Java.

String Buffer is synchronised: multiple thread cannot be executed simultaneously.

Thread safe class.

String Builder:

String Builder is present in java 5.

String Builder is asynchronous: multiple thread can be executed simultaneously.

Not thread safe class.

6.How to reverse a string with & without inbuild method.

With Inbuild Method:

class HelloWorld {

public static void main(String[] args) {

StringBuffer s=new StringBuffer("Hello");

System.out.println(s.reverse());

}

}

Without Inbuild Method:

class HelloWorld {

public static void main(String[] args) {

String a="Hello";

for(int i=a.length()-1;i>=0;i--){

System.out.print(a.charAt(i));

}

}

}

7.OOPS Concept in JAVA.

OOPS is bind together the data and function can operate them, no other part of the code can access them, Except the function.

Realtime Methodolgy.

Code Reusability.

Security.

Optimized Code.

OOPS Concepts are:

1. class
2. object
3. encapsulation
4. polymorphism
5. Inheritance
6. Data Abstraction

Composition:

* One object has a relationship with another object.
* It is Lousily coupled.
* Has a relationship.
* Can be changed.
* E.g. (car has a battery).

Class:

* Collection of objects.
* Template of an object

Object:

* It is a state and behaviour of class.

Encapsulation:

* Hiding and Binding of data into single unit
* Getter & Setter Method.
* Constructor & to String.

Constructor:

1. Is a special type of method.
2. To initialize instance variable of object.
3. It must be same name as class name.
4. It has no return type.

Constructor has two type .

* Default Constructor.
* Parameterized Constructor

Inheritence:

1. A class inherit all the properties from another class by using extends keyword.
2. Is a relationship (tightly Coupled).
3. There are 5 type of Inheritence:

* Single Inheritence:

One child class inherits all the property from another parent class.

E.g. .(Car->Sports Car)

* Multi-level Inheritence :

One child class act as apparent to another child class.

E.g. .(Grandfather(Parent Class)->Father(act as Parent Class)->child(Child Class))

* Hierarchical Inheritence :

One parent class has more than one child class.

E.g.(A(Parent Class)->b,c,d(Child Class))

* Multiple Inheritence :

One child has more than one parent class

->We can’t achieve this in inheritance .

->We can achieve it through Interface.

E.g(a,b(Parent class)->c(Child class))

* Hybrid Inheritance:

It is a diamond like structure and it is a combination of Hierarchical and Multilevel Inheritence.

->We can’t achieve this in inheritance .

->We can achieve it through Interface.

Polymorphism:

Poly->Many, Morphism ->Forms

It is used to achieve different functionality in same function name.

There are 2 types of polymorphism:

->Run-time Polymorphism (Method Overriding) or (Dynamic Binding).

->Compile time Polymorphism (Method Overloading) or (Static Binding).

Method Overriding:

Child class as same method has parent class but only different in implementation.

Without Inheritance we can’t achieve method overriding.

Method name Should be same

Parameter should be same

Return type should be same

But different in implementation class.

Method Overriding:

A class multiple method with same method name but different in parameter and return type.

Same method name in same class but different in parameter and return type.

Data Abstraction:

->Process of hiding inner implementation & show only external functionality.

It can achieve by 2 ways:

1. Interface.
2. Abstract class.

Interface:

* Used to show external functionality.
* Blueprints of methods.
* It contains unimplemented method we can implement method.
* Can’t create object for interface.
* Used to achieve hybrid inheritance.

Abstract Class:

* It contains both implemented and unimplemented method.
* Can’t create object for abstract.
* Extend keyword is used.
* Abstract keyword to declare class name.
* Unimplemented method needs to have abstract keyword.

Syntax:

Abstract class name{

}

Class -> Class => Extends

Interface -> class => Implements

abstract class -> class => extends

interface -> abstract class => implement

interface -> interface => extends

8.Access Specifiers:

* **public:** Accessible in all classes in your application.
* **protected:** Accessible within the package in which it is defined and in its **subclass.**
* **private:** Accessible only within the class in which it is defined.
* **default (declared/defined without using any modifier):** Accessible within the same class and package within which its class is defined.

9. Different Between Collection & Collections.

| **Collection** | **Collections** |
| --- | --- |
| It is an interface. | It is a utility class. |
| It is used to represent a group of individual objects as a single unit. | It defines several utility methods that are used to operate on collection. |
| concept for storing data, rather than a tool for manipulating data | NullPointerException if the collections or class objects provided to them are null. |
| The Collection is an interface that contains a static method since java8. The Interface can also contain abstract and default methods. | It contains only static methods. |

10.Diffrence between List, Set & Map

List:

To Store Large Amount of a data and manipulate.

Dynamic size

Allows Duplicate values.

It allows null values.

It follows insertion Order.

Default size is 10.

Set:

It doesn’t allow duplicate values.

It allows only one null value.

It doesn’t follow insertion order.

Map:

It stores values in the form of key and values.

Key should be unique.

Values can have duplicate.

It allows only one null key but it allows multiple null values.

It doesn’t follow insertion order.

It defaults size is 10.

11.Difference between Array & Array List

Array:

Collection of similar Data

Data type can be in primitive /object.

Arr.length

Fixed Size

Array List:

Collection of similar Data

Object only can be stored.

Arr.size()

Dynamic size

12.Hashing Technique:

It takes the object & convert into integer based on hash code formula.

Index=hash % n.

Hash Table:

Hashing is used to breakup huge amount of data into small table.

It helps to reduce the searching time & space.

Hash Bucket:

It stores one or more object which follows linked list.

Hash Collision:

It happens when the index value is same

To avoid the collision, it checks with equals method. If both values are same it will overwrite otherwise it will create a node and store the value.

Hash Code:

It returns integer hash code for values it used to find right bucket for object.

Equals():

It used to compare the values of an object to avoid collision.

13.Functional Interface or Sam Interface

It is also known as Sam interface. (SAM- Single Abstract Method)

It exactly contains only one unimplemented method.

We can have n no static & default method in interface.

Another way to declare functional Interface is by using @Functional Interface Annotation

Functional Interface can be implemented in two ways.

-Lambada Expression.

-Implements.

14.Four Pre defined functional interface.

Consumer.

Predicate.

Function.

Supplier.

15.Lambda Expression

It is short block of code which take input as parameter and return value.

It is an anonymous function.

It is used to implement the method of functional interface.

Thread:

It represents a separate flow of execution within a program.

It allows program to perform multiple operation simultaneously.

Thread can be created in 2 ways:

-By extending thread class

-By implementing runnable interface.

If I extend thread, it has all properties of thread.

public class Thread1 extends Thread{

public void run() {

System.***out***.println("Thread is running");

}

public static void main(String[] args) {

Thread1 t1=new Thread1();

t1.start();

System.***out***.println("Main Thread is Runnig");

}

}

If I implement is runnable it doesn’t have the property of thread so we need to create object for thread.

public class RunThread implements Runnable {

*@Override*

public void run() {

System.***out***.println("Thread is Running");

}

public static void main(String[] args) {

RunThread r1=new RunThread();

Thread t1=new Thread(r1);

t1.start();

System.***out***.println("Main Thread");

}

}