NAME : Arunkumar.B.jaiswal.

CCPP ID : MD0171.

1. What is the difference between string , string builder ,string buffer.

* STRING
* String is ***immutable*** (once created cannot be changed ) object.
* Object created as a string is stored in constant string pool.
* Every immutable object in java is **thread safe ,** that implies string is also thread safe ,so no two threads can be used by string simultaneously.
* String once assigned cannot be changed.
* STRING BUFFER
* String buffer is **synchronized** i.e **thread** **safe**.
* So two threads can’t call the method of string buffer simultaneously.
* It is less efficient then string builder.
* String buffer is ***mutable***, so the value of an object can be changed.
* String buffer can be converted to string by using **toString**() method.
* STRING BUILDER
* String buffer is **not** synchronized i.e **not** thread safe.
* So two threads can call the method of string buffer simultaneously.
* It is much way better than string buffer.
* String buffer is ***mutable***.

1. Can we overload and override static methods in java?

* **Overload** ( same name , but different signature i.e input parameter or type of parameter , compile time polymorphism )
* Yes we can overload two static methods with same name , but differences in input parameter.

e.g :

 public static void food(){}

public void food(100){}

* **Note :-**  But we cannot overload two methods in java if they differ only by static keyword (assuming that no. of parameters and types of parameters is same).
* If the signature is different overloading can be done but if the difference in signature is based out of static keyword then overloading can’t be done.

e.g :

 public static void food(){}

public void food(){} // compile error

* **Override**( same name , same signature , runtime polymorphism)
* NO , we can declare static method with same signature in subclass, but it is not considered overriding as there won’t be any runtime polymorphism. Its actually method Hiding.

e.g :

class Base {

// Static method in base class which will be hidden in subclass

public static void display() {

System.out.println("Static or class method from Base");

}

}

class Derived extends Base {

// This method hides display() in Base

public static void display() {

System.out.println("Static or class method from Derived");

}

}

}

public class Test {

public static void main(String args[ ]) {

Base obj1 = new Derived();

// As per overriding rules this should call to class Derive's static

// overridden method. Since static method can not be overridden, it

// calls Base's display()

obj1.display(); // Static or class method from Base

}

}

If the reference is of base type ,then derive’s static overridden method never gets called therefore , static method can not be overridden.

1. Why main is public static and void in java?

* ***PUBLIC***: If the Access specifier is public we can call that class or method from anywhere, so when we install java in our system, java files is in some folder or package for example in download folder and our program is in another xyz folder so for JVM to call from one folder to our respective folder our access specifier needs to be public.

* ***STATIC***: Benefit of static keyword is that while calling object we did not need to create object of that class. So JVM do not need to create object of the class where is main method.
* ***VOID***: we do not want that our method returns some kind of value to JVM because this method gets called by JVM.

So above all where the reasons main method is public, static as well as void in java.

1. Is java pure object oriented programming? If yes then why if no they why?

* For a language to be fully ***object oriented programming*** it should follow given certain requirements.
* Encapsulation/data Hiding
* Inheritance
* Polymorphism
* Abstraction
* All predefined are objects
* All user defined should be objects.
* All operations performed on objects must be through methods exposed at the objects.
* A purely object oriented programming doesn’t support primitive data type like int,char,float …etc.
* Java uses **wrapper** classes which provides conversion of primitive type to object type which internally uses boxing and unboxing which also should be avoided.
* Java also uses arithmetic operators ‘ + ‘ which is not a case in pure oop.
* So , considering above conditions java is not a purely object oriented programming though it uses wrapper class.

1. What is difference between abstract and interface class?

* ABSTRACT
* An abstract class can extend another java class and can implements multiple java interfaces.
* It doesn’t support multiple inheritance.
* Abstract class can have abstract and non abstract methods.
* It can have final, non final, static, non static variables.
* The abstract keyword is used to declare abstract class.
* INTERFACE
* An interface can extend another java interface only.
* It supports multiple inheritance.
* Interface can only have abstract methods.
* Interface only has static and final variables.
* The interface keyword is used to declare interface.
* Interface is not a class so we saved extends keyword.

1. If two threads are running simultaneously then how you will stop one and let the other one first finish that then start that one?

* JOIN()
* The **join**() method waits for a thread to die. i.e ,it causes the currently running threads to stop executing until the thread it joins with complete its tasks.
* It throws **Interrupted** Exception.

e.g:

public static void main( String [] args){

Thread t1= new Thread();

t1.start();

t1.join(); // main method thread is executing this line so there is a contract between main thread and t1 so now main thread will further doesn’t execute the line until the operation of thread t1 is completed.

Thread t2= new Thread();

T2.start();

}

1. What is the difference between comparable and comparator?

Java provides two interface to sort objects using data members of the class:

* Comparable.
* Comparator.
* COMPARABLE:
* It is a default natural sorting order.
* It is present in java.lang package.
* It has only 1 method i.e compareTo().
* All wrapper class and string class implements comparable.
* It provides a single sorting sequence , we can sort the collections on the basis of a single element such as id,name and price.
* COMPARATOR:
* It is a customized sorting order.
* It is in java.util package.
* There are 2 methods compare() and equals() , but we only use compare().
* It provides multiple sorting sequence.

1. What is the difference between iterator and iterable?

* ITERATOR:
* It is one of the 3 cursors of java.
* If we want to retrieve objects one by one from collections ,then we should go for cursor.
* Iterator is consider as a universal cursor, it is way much better than enumeration and it has ability to read as well as remove.
* Signature: public Iterator iterator()
* Its has 3 methods .
* The problem with iterator is that it can move in forward direction only and not in reverse direction.
* ITERABLE:

1. How can you iterate over a list? Different ways to iterate it?
2. Is java platform dependent or independent language? Why?

* YES, Java programming language is platform independent and not JVM because for different architecture different JVM is initialized.
* The byte code which is generated by JVM for every program can be run on any operating system that makes the java language independent.
* Where as for different machine different JVM is initialized.