1)What is String constant pool?Why string is immutable?difference between String,StringBuffer,StringBuilder?Difference between == operator and .equals() method?Explain the diffrence between jdk,jre,jvm?why java is platform independent?

Answer:

If we assign the value to the String literals then the object will be created in the String constatnt pool.

String is immutable because if we modify the existed the string object then the new object will be created in the string pool area, in the string pool area its not possible to make the reference.

String vs StringBUffer vs String Builder:

String is immutable and its non synchronized.

StringBuilder is mutable and its synchronized.The performance will be high than comapre to StringBUffer.

StringBUffer is mutuable and its non synchronized.The performance will be less than compare to the StringBuilder.

== operator vs .equals():

== operator is used to the check the refrence of the variable.

.equals will be used to check the content of the variable.

**public** **class** DoubleEqualsOperator

{

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

{

String name="Welcome";

String name1=**new** String("Welcome");

System.***out***.print(name==name1);//false

System.***out***.print(name.equals(name1));//true

}

}

JVM vs JRE vs JDK

JVM means Java virtual mechine, its not physically existed it is used to convert Byte code &

then it will convert byte code instructions in to understandable format of operating system.

JRE means Java Runtime Environment, its physically existed it is used to developing the java applications and it contains libraries and other files that JVM uses at runtime.

JDK means java develoment kit, its used to develop java applications and applets.it contains JRE+development tools.

Platform independent:

Suppose if you write the java source code in one operating system then we will compile that we will get .class file. If we copy the .class file any of the operating systems like linux, window, its used to run the file.so, That’s why java is platform independent.

2)Difference between Class and Object?Explain in detail OOP's Concept?Explain static and dynamic polymorphism with examples?Difference between static and instance method with examples?Difference between this and super?Difference between method and constructor? Difference interface and Abstract class?What is the significance of public,private,protected classes?

Class vs object

class is like an idea of the object.

Object will be created from the class.

OOPS concepts:

1.Inheritance

2.Polyorphism

3.Encapsulation.

4.DataHiding.

1.Inheitance:

Inheritance is the process of acquiring the properties from parent clas to the child class by using the extends kyword.

The Inheritance will be sub classified in to the 5 types:

1.Single level Inheritance.

2.Multilevel inheritance.

3.Hierarchial inheritance.

4.Multiple inheritance.

5.Hybrid inheritance.

1.Single level inheritance: In the single level inheritance the properties will be acquiring from parent class to the child class.

2.Multilevel inheritance: in the mutilevel inheritance the the properties will acuquiring from parent class to the child class and child class to the grand child class.

3.Hierarchial inheritance: in this inheritance the properties will be inherited from parent class to more than one child class is nothing but the hierarchial inheritance.

4.Multi level inherutance: in this inheritance the properties will be acquiried from more than one parent class to the child class.

5.Hybrid inherutance: it is the combination of any of two inheritances from the above listed.

2.Polymorphism: it is the process of converting one form in to different forms.

Polymorphism is the combination of Method over loading and method overriding.

Method overloading: in the method overloading the method name should be same and the arguements will be different.

Method overriding: in the method method overriding the method name, paramters and return should be same.

3.Encapsulation: Encapsulation is the process of binding data and corresponding methods in to a single unit is know an Encapsulation. In the encapsulation we are using the setter and getter methods to cahng

4.DataHiding:data hiding is the process of hiding the sensitive data using the private access modifier.

Static polymorphism: It also caled as method over loading or early binding, here we can took the same method anme with different parameters/Arguements.

**import** java.util.Scanner;

**public** **class** StaticPolymorphism

{

**void** Toucan(String name)

{

System.***out***.println("My name is:"+name);

}

**void** Toucan(**long** salary)

{

System.***out***.println("MY salary is:"+salary);

}

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

{

Scanner sc = **new** Scanner(System.***in***);

StaticPolymorphism SP =**new** StaticPolymorphism();

System.***out***.println("Enter you name:");

SP.Toucan(sc.nextLine());

System.***out***.println("Enter you salary:");

SP.Toucan(sc.nextLong());

}

}

Dynamic polymorphism:IT is also called as method overriding or late binding, here we can took the same method name, same return type and same parameters.

**public** **class** DynamicPolymorphismParent

{

**void** asset(**long** value)

{

System.***out***.print("Parent asset:"+value);

}

}

**import** java.util.Scanner;

**public** **class** DynamicPolymorphismChild **extends** DynamicPolymorphismParent

{

**void** asset(**long** value)

{

System.***out***.print("child asset:"+value);

}

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

{

Scanner sc = **new** Scanner(System.***in***);

DynamicPolymorphismChild dpc = **new** DynamicPolymorphismChild();

dpc.asset(sc.nextLong());

}

}

static vs instance method:

static methods are methods in java which we can call by using the method name itself.

**import** java.util.Scanner;

**public** **class** StaticMethod

{

**static** **void** person(String name)

{

System.***out***.print("Name:"+name);

}

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

{

Scanner sc = **new** Scanner(System.***in***);

*person*(sc.next());

}

}

Instance methods is a method in java which we can create the oboject before calling the method, the method should be called by using object.

**import** java.util.Scanner;

**public** **class** InstanceMethod

{

**void** vehicle(String name)

{

System.***out***.print("Vehicle name:"+name);

}

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

{

Scanner sc = **new** Scanner(System.***in***);

InstanceMethod i = **new** InstanceMethod();

i.vehicle(sc.next());

}

}

this vs super:

this keyword mainly represents the current instance of a class.

this keyword used to call default constructor of the same class.

super keyword represents the current instance of a parent class.

Method vs constructor:

A method is a block of code which only runs when it is called.whereas method is used to exhibits functionality of an object.whereas methods are invoked explicitly. where the method may/may not return a value.

Constructor is used to initialize an object.Constructors are invoked implicitly.

Constructor does not return any value.

Interface class vs Abstract class:

in the interface class we are written the unimplemented methods only. The interface contains only Abstract methods, if we want to write an implemented methods we can use static or default keyword.

But in Abstract class we are written the Both implemented and un impleneted methods in a single class, if the method is not implemented we are using the Abstract keyword.

Public: The access level of public modifier,we can access any where from the project.

[private: The](private:THe) access level of private modifier will be with in the class only.

Protected: The access level of protected modifier will be with in the package and outside of the package through child class.

3)Difference between Checked and unchecked exceptions?Give 1 example on each?Difference between final,finally,finalize()?Give 1 example on each?Difference between throw and throws?Give 1 example on each?Write a code to show NullPointerException?

Checked exceptions: The exception will be checked at the compiler.

Unchecked exceptions: The exceptions are not checked at the compiler.

**public** **class** NullpointerExceptions

{

String name=**null**;

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

{

**try** {

System.***out***.print(name);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

final vs finally vs finalize:

final is the modifier applicable for classes, methods, variables.

Finally is the block always associated with try – catch to maintain cleanup code which should be executed always irrespective of whether exception raised or not raised and whether handled or not handled

finalize is the method always invoke by garbage collector just before destroying an object to perform cleanup activities.

**final** **class** Final

{

**void** display()

{

System.***out***.print("raja");

}

}

**public** **class** Finalmain **extends** Final

{

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

{

Finalmain f = **new** Finalmain();

f.display();

}

}

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** Finally

{

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

{

**try** {

FileWriter fw = **new** FileWriter("abc.txt");

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**finally**

{

System.***out***.print("Raja");

}

}

}

Throw vs Throws:

Throw is also a predefined keyword, the main purpose of this keu=yword to raise the custom defined exceptions.

We have to use throw keyword inside method body.

When ever we use throw keyword inside method body to raise custom defined exceptions, then we must need to use throws at method heading.

Throws ia also a predefined keyword by using this we can able to hanle checked exceptions but not unchecked exceptions without using try & catch block.

By using this keyword we can not provide any user friendly error message to the user.

**public** **class** TooOldException **extends** RuntimeException

{

TooOldException (String msg)

{

**super**(msg);

}

}

**public** **class** TooYoungException **extends** RuntimeException

{

TooYoungException(String msg)

{

**super**(msg);

}

}

**import** java.util.Scanner;

**public** **class** customizedException

{

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

{

Scanner sc = **new** Scanner(System.***in***);

**int** age=sc.nextInt();

**if**(age<18)

{

System.***out***.println("ggy");

**throw** **new** TooYoungException("u r not eligible");

}

**else** **if**(age>60)

{

**throw** **new** TooOldException("u r too late");

}

**else**

{

System.***out***.print("Your eligible for the marriage");

}

}

}

import java.io.FileWriter;

import java.io.IOException;

public class ThrowsException {

private String a;

public void meth1(String a)

{

this.a="raja";

}

public static void main(String[] args) throws IOException

{

FileWriter se = new FileWriter("abc.txt");

try

{

System.out.print(10/0);

}

catch(Exception e)

{

}

finally

{

se.close();

}

}

}

Nullpointer Exception:

**public** **class** NullpointerExceptions

{

String name=**null**;

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

{

**try** {

System.***out***.print(name);

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

4)How many ways we can create a thread?best approach what?Difference between sleep and wait methods?Explain difference methods present in Thread class?How to Create a Deadlock Scenario Programatically?Explain THread life cycle with all methods?Explain synchronization?Difference between static and synchronized block?

We can create the thread by

1.By using extends method

2.By using runnable interface

The best approach is by using runnable interface. By using this method we can acquire the properties from than one class.

Sleep vs Wait:

|  |  |
| --- | --- |
| Wait() method belongs to Object class. | Sleep() method belongs to Thread class. |
| Wait() method releases lock during Synchronization. | Sleep() method does not release the lock on object during Synchronization. |
| Wait() should be called only from Synchronized context. | There is no need to call sleep() from Synchronized context. |

difference methods present in Thread class:

Thread class provides various static methods that are as follows:

1. currentThread(): The currentThread() returns the reference of currently executing thread. Since this is a static method, so we can call it directly using the class name. The general syntax for currentThread() is as follows:

Syntax:

public static Thread currentThread()

2. sleep(): The sleep() method puts currently executing thread to sleep for specified number of milliseconds. This method is used to pause the current thread for specified amount of time in milliseconds.

Since this method is static, so we can access it through Thread class name. The general syntax of this method is as follows:

Syntax:

public static void sleep(long milliseconds) throws InterruptedException

The general syntax for overloaded version of sleep method is as follows:

Syntax:

public static void sleep(long milliseconds, int nanoseconds ) throw InterruptedException

The overloaded version of sleep() method is used to pause specified period of time in milliseconds and nanoseconds. Both methods throw InterruptedException and must be used within Java try-catch block.

3. yield(): The yield() method pauses the execution of current thread and allows another thread of equal or higher priority that are waiting to execute. Currently executing thread give up the control of the CPU. The general form of yield() method is as follows:

Syntax:

public static void yield()

4. activeCount(): This method returns the number of active threads.

Syntax:

public static int activeCount()

Since this method is static, so it can be accessed through Thread class name. It does not accept anything.

The instance methods of Thread class are as follows:

1. start(): The start() method is used to start the execution of a thread by calling its run() method. JVM calls the run() method on the thread. The general syntax for start() method is as follows:

Syntax:

public void start()

2. run(): The run() method moves the thread into running state. It is executed only after the start() method has been executed. The general syntax is as follows:

Syntax:

public void run()

3. getName(): This method returns the name of the thread. The return type of this method is String. The general form of this method is:

Syntax:

public final String getName()

4. setName(): This method is used to set the name of a thread. It takes an argument of type String. It returns nothing.

Syntax:

public final void setName(String name)

5. getPriority(): This method returns the priority of a thread. It returns priority in the form of an integer value ranging from 1 to 10. The maximum priority is 10, the minimum priority is 1, and normal priority is 5.

Syntax:

public final int getPriority() // Return type is an integer.

6. setPriority(): This method is used to set the priority of a thread. It accepts an integer value as an argument. The general syntax is given below:

Syntax:

public final void setPriority(int newPriority)

7. isAlive(): This method is used to check the thread is alive or not. It returns a boolean value (true or false) that indicates thread is running or not. The isAlive() method is final and native. The general syntax for isAlive() method is as follows:

Syntax:

public final native boolean isAlive()

8. join(): The join() method is used to make a thread wait for another thread to terminate its process. The general syntax is

Syntax:

public final void join() throw InterruptedException

This method throws InterruptedException and must be used within a try-catch block.

9. stop(): This method is used to stop the thread. The general form for this method is as follows:

Syntax:

public final void stop()

This method neither accepts anything nor returns anything.

10. suspend(): The suspend() method is used to suspend or pause a thread.

Syntax:

public final void suspend()

11. resume(): This method is used to resume the suspended thread. It neither accepts anything nor returns anything.

Syntax:

public final void resume()

12. isDaemon(): This method is used to check the thread is daemon thread or not.

Syntax:

public final boolean isDaemon()

Synchronization in java is the capability to control the access of multiple threads to any shared resources. Java synchronization is better option where we want to allow only one thread to acces the share resources.

Static vs synchronization block:

Synchronized block can be used to perform synchronization on any specific resource of the method.

Suppose we have 50 lines of code in our method, but we want to synchronize only 5 lines, in such cases, we can use synchronized block.

If we put all the codes of the method in the synchronized block, it will work same as the synchronized method.

5)Internal working of HashMap?What is ConcurrentHashMap give one example program? when we go for list ,set,map?Frequent operation is searching what we need to use?

HashMap: It is the process of converting oject class in to integer. The integer value indexing the faster searching.

HashMap is also the part of collection framework.it uses a technique is called hashing.it implements the map interface.it stores the value in the form of keys & values. The values will be arranged in the form of Hashcode values that will be store in to the Hashmap.

The followig are trhe different methods in HashMap:

equals(): It checks the equality of two objects. It compares the Key, whether they are equal or not. It is a method of the Object class. It can be overridden. If you override the equals() method, then it is mandatory to override the hashCode() method.

hashCode(): This is the method of the object class. It returns the memory reference of the object in integer form. The value received from the method is used as the bucket number. The bucket number is the address of the element inside the map. Hash code of null Key is 0.

Buckets: Array of the node is called buckets. Each node has a data structure like a LinkedList. More than one node can share the same bucket. It may be different in capacity.

What is ConcurrentHashMap give one example program:

 This class obeys the same functional specification as Hashtable and includes versions of methods corresponding to each method of Hashtable. However, even though all operations are thread-safe, retrieval operations do not entail locking, and there is not any support for locking the entire table in a way that prevents all access. This class is fully interoperable with Hashtable in programs that rely on its thread safety but not on its synchronization details

when we go for list ,set,map::

list: if we want to represent the elements in the form of list we can use the list and it allows the duplicate elements.

Set:if we want to want to represents the elelments in the form of set we can use the set and it allows only unique elements.

Map:if we want to represents the elements in the form of keys and values we can use this map interface.

Frequent operation is searching what we need to use:

For searching of the elements we are using ArrayList, by using ArrayList we can searching the objects by using the index value.

6)Explain java 8 features?

1.Lambda expressions:

Lambda expression helps us to write our code in functional style. It provides a clear and concise way to implement SAM interface(Single Abstract Method) by using an expression. It is very useful in collection library in which it helps to iterate, filter and extract data.

2.Method references:

Java 8 Method reference is used to refer method of functional interface . It is compact and easy form of lambda expression. Each time when you are using lambda expression to just referring a method, you can replace your lambda expression with method reference.

3.Functional interfaces:

An Interface that contains only one abstract method is known as functional interface. It can have any number of default and static methods. It can also declare methods of object class.

Functional interfaces are also known as Single Abstract Method Interfaces (SAM Interfaces).

4.Collectors class.

Collectors is a final class that extends Object class. It provides reduction operations, such as accumulating elements into collections, summarizing elements according to various criteria etc.

5.Stream API:

Java 8 java.util.stream package consists of classes, interfaces and an enum to allow functional-style operations on the elements. It performs lazy computation. So, it executes only when it requires.

6.ForEach() method:

Java provides a new method forEach() to iterate the elements. It is defined in Iterable and Stream interfaces.It is a default method defined in the Iterable interface. Collection classes which extends Iterable interface can use forEach() method to iterate elements.

This method takes a single parameter which is a functional interface. So, you can pass lambda expression as an argument.

7.Single Filter:

Java stream provides a method filter() to filter stream elements on the basis of given predicate. Suppose, you want to get only even elements of your list, you can do this easily with the help of filter() method.

7)Print the nearest prime to given number

Input:10

Output:7,11//10 below nearest palindrome 7 and 10 above 11

Answer:

**import** java.util.Scanner;

**public** **class** NearestPrimeNumber

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number:");

**int** num=sc.nextInt();

**int** dcount=0;

**int** icount=0;

**int** fcount=0;

**int** i;

**int** k;

**for**(i=num; i>=1; i--)

{

dcount++;

fcount=0;

**for**(**int** j=1; j<=num; j++)

{

**if**(i%j==0)

{

fcount++;

}

}

**if**(fcount==2)

{

**break**;

}

}

**for**(k=num+1; ; k++)

{

icount++;

fcount=0;

**for**(**int** j=1; j<=num; j++)

{

**if**(k%j==0)

{

fcount++;

}

}

**if**(fcount==2)

{

**break**;

}

}

**if**(dcount<icount)

{

System.***out***.print(i);

}

**else** **if**(icount==dcount)

{

System.***out***.print(i+" "+k);

}

**else**

{

System.***out***.print(k);

}

}

}

8.How to swap two numbers without using a third variable?

Answer:

Example:

a=a+b-(b=a); here, b value is stored in to the a & a value is stored in to the b.

9)Reverse Number and String program

Input: 123

output:321

Input:coachit

Output:tihcaoc

Answer:

**import** java.util.Scanner;

**public** **class** ReverseNumberAndString

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the Number:");

**int** num=sc.nextInt();

**int** rnum=0;

**while**(num>0)

{

rnum=rnum\*10+num%10;

num=num/10;

}

System.***out***.print(rnum);

System.***out***.print("\nEnter the name:");

String name=sc.next();

String rename=" ";

**char** ch;

**for**(**int** i=0; i<name.length(); i++)

{

ch=name.charAt(i);

rename=ch+rename;

}

System.***out***.print(rename);

}

}

10)Fibonacci Series program?Find factorial of an integer?

Answer:

**import** java.util.Scanner;

**public** **class** FibonacciSeriesProgram

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number of Fibonacci values:");

**int** ivalue=0;

**int** svalue=1;

**int** tvalue;

**int** num=sc.nextInt();

System.***out***.print("Fibonacci values:");

**for**(**int** i=0; i<num; i++)

{

System.***out***.print(ivalue+" ");

tvalue=ivalue+svalue;

ivalue=svalue;

svalue=tvalue;

}

}

}

**import** java.util.Scanner;

**public** **class** FactorialOfANumber

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number:");

**int** num=sc.nextInt();

System.***out***.print("Factorials of a number");

**for**(**int** i=1; i<=num; i++)

{

**if**(num%i==0)

{

System.***out***.print(i+" ");

}

}

}

}

11)List contains some Numbers?Print only even numbers from the list?

Answer:

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** ListContainsSomeNumbersPrintEvenNumbers

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the size of the Array:");

**int** size=sc.nextInt();

**int** []Array = **new** **int**[size];

System.***out***.print("Arraylist elements:");

**for**(**int** i=0; i<Array.length; i++)

{

Array[i]=sc.nextInt();

}

ArrayList<Integer> num = **new** ArrayList<Integer>();

System.***out***.print("Even Numbers from ArrayList:");

**for**(**int** i=0; i<Array.length; i++)

{

**if**(Array[i]%2==0)

{

num.add(Array[i]);

}

}

System.***out***.print(num);

}

}

12)Sorting an array in Java?

input:1, 2, 3, -1, -2, 4

output:-2,-1,1,2,3,4 do without using special methods and with special methods?Find second largest number in the array?Check particular elements present in array or not?

Answer:

Soring of an array by using special methods.

**public** **class** SortingonAnArray

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the size of the array:");

**int** size = sc.nextInt();

**int** []Num=**new** **int**[size];

System.***out***.print("Array Elements:");

**int** i;

**for**( i=0; i<Num.length; i++)

{

Num[i]=sc.nextInt();

}

System.***out***.print("Sorting of Array Elements:");

**for**( i=0; i<Num.length; i++)

{

Arrays.*sort*(Num);

System.***out***.print(Num[i]+" ");

}

}

}

second largest number in the array:

**public** **class** SortingonAnArray

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the size of the array:");

**int** size = sc.nextInt();

**int** []Num=**new** **int**[size];

System.***out***.print("Array Elements:");

**int** i;

**for**( i=0; i<Num.length; i++)

{

Num[i]=sc.nextInt();

}

System.***out***.print("\nSecond largest number in an array:");

**for**(**int** i1=0; i1<Num.length; i1++)

{

**if**( i1==Num.length-2)

System.***out***.print(Num[Num.length-2]+" ");

}

}

}

check particular elements present in array or not:

**public** **class** SortingonAnArray

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the size of the array:");

**int** size = sc.nextInt();

**int** []Num=**new** **int**[size];

System.***out***.print("Array Elements:");

**int** i;

**for**( i=0; i<Num.length; i++)

{

Num[i]=sc.nextInt();

}

**int** target=234;

**for**(**int** j=0; j<Num.length; j++)

{

**if**( target==Num[j])

{

System.***out***.print("The Element is present in the Array");

**break**;

}

**else**

{

System.***out***.print("The element is not present in theArray");

**break**;

}

}

}

}

13)Check if two arrays contains same elements or not?

Answer:

**public** **class** TwoArraysContainsSameElement

{

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

{

**int**[]list1 = {1,2,3,4,5};

**int**[] list2 = {1,2,3,4,5,6};

**int** count=0;

**for**(**int** i=0; i<list1.length; i++)

{

**for**(**int** j=0; j<list2.length; j++)

{

**if**(list1[i]==list2[j])

{

count++;

}

}

}

**if**(count==list1.length)

{

System.***out***.print("The arrays contains common numbers");

}

**else**

{

System.***out***.print("The arrays contains different numbers");

}

}

}

14)Check given Number is perfect or not

Input: 6 (6 is divisable by 1,2,3)1+2+3

Output:true

**import** java.util.Scanner;

**public** **class** NumerIsPerfect

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number:");

**int** num=sc.nextInt();

**int** sum=0;

**for**(**int** i=1; i<num; i++)

{

**if**(num%i==0)

{

sum=sum+i;

//System.out.print(i+" ");

}

}

//System.out.print(sum);

**if**(sum==num)

{

System.***out***.print("perfect number");

}

**else**

{

System.***out***.print("its not a perfect number");

}

}

}

15)Amstrong Number program

input:153(1**1**1+5**5**5+3**3**3=153)

output:true

Answer:

**import** java.util.Scanner;

**public** **class** ArmStrongNumber

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the number:");

**int** num=sc.nextInt();

**int** temp=num;

**int** renum=0;

**int** sum=0;

**int** count=0;

**while**(num>0)

{

num=num/10;

count++;

}

//System.out.print(count);

num=temp;

**while**(num>0)

{

renum=num%10;

num=num/10;

sum=sum+(**int**)Math.*pow*(renum,count);

}

//System.out.print(sum);

**if**(sum==temp)

{

System.***out***.print("true");

}

**else**

{

System.***out***.print("false");

}

}

}

16)find the largest and smallest element in the array?

Answer:

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** LargestAndSmallestElementInAnArray

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the size of the Array:");

**int** size=sc.nextInt();

**int**[]num=**new** **int**[size];

System.***out***.print("Array elements:");

**for**(**int** i=0; i<num.length; i++)

{

num[i]=sc.nextInt();

}

Arrays.*sort*(num);

System.***out***.print("\nSmallest element in an array:"+num[0]);

System.***out***.print("\nLargest element in an array:"+num[num.length-1]);

}

}

17)find the number of vowels and consonants count in the String

Input:ramesh

output:vowels:2,

consonants:4

Answer:

**import** java.util.Scanner;

**public** **class** ConsonantsAndVowelsInAnArray

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the name:");

String Name=sc.next();

**int** Vcount=0;

**int** Ccount=0;

String name=Name.toLowerCase();

**for**(**int** i=0; i<name.length(); i++)

{

**if**(name.charAt(i)=='a'||name.charAt(i)=='e'||name.charAt(i)=='i'||name.charAt(i)=='o'||name.charAt(i)=='u')

{

Vcount++;

}

**else** **if**(name.charAt(i)<='a' || name.charAt(i)<='z' )

{

Ccount++;

}

}

System.***out***.print("\nConsonants Count:"+Ccount);

System.***out***.print("\nVowels Count:"+Vcount);

}

}

18)word reverse and First letter capital program

input:this is java

output:Siht Si Avaj

Answer:

**import** java.util.Scanner;

**public** **class** ReverseWordAndFirstLetterCapital

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter the name:");

String name=sc.nextLine();

String word ;

String rword=" ";

String rword1=" ";

String []Name =name.split(" ");

System.***out***.print("Reverse of a string:");

**for**(**int** i=0; i<Name.length; i++)

{

word = Name[i];

rword=" ";

**for**(**int** j=0; j<word.length(); j++)

{

**char** ch=word.charAt(j);

rword = ch+rword;

}

rword1=rword1+rword;

}

System.***out***.println(rword1);

String cname=" ";

String []CName =name.split(" ");

**char** ch1;

System.***out***.print("Reverse of a string and First letter should be capital:");

**for**(**int** i=0; i<CName.length; i++)

{

word = CName[i];

rword=" ";

**for**(**int** j=0; j<word.length(); j++)

{

**if**(j==word.length()-1)

{

ch1 = Character.*toUpperCase*(word.charAt(j));

}

**else**

{

ch1=Character.*toLowerCase*(word.charAt(j));

}

rword = ch1+rword;

}

System.***out***.print(rword);

}

}

}

19)Remove space in a String

input:this is java

Output:thisisjjava

Answer:

**import** java.util.Scanner;

**public** **class** RemoveSpaceInString

{

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

{

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter name:");

String name=sc.nextLine();

String []Name =name.split(" ");

**for**(**int** i=0; i<Name.length; i++)

{

System.***out***.print(Name[i]);

}

}

}

20)List is the collection of person object Person class contains name,age From the list of person object remove below 18 age person objects after removing print latest List of person obect? Do with and without stream api?

Answer:

**import** java.util.\*;

**import** java.util.stream.Collectors;

**public** **class** Person

{

String name;

**int** age;

Person(String name, **int** age)

{

**this**.name=name;

**this**.age=age;

}

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

{

List <Person>person = **new** ArrayList<Person>();

person.add(**new** Person("Rama",19));

person.add(**new** Person("Raja", 15));

person.add(**new** Person("Mounika",24));

person.add(**new** Person("Akshata", 19));

person.add(**new** Person("Sanjay",20));

person.add(**new** Person("Chandu", 17));

person.stream().filter(P -> P.age >18).collect(Collectors.*toList*()).forEach(Person ->System.***out***.print(Person.name+" "));

}

}

21)Example program on comparable and comparator?In which scenario we use comparator?

Answer:

comparable provides a single storing sequence.Like by (age, id, name, price) it provides compareTo() method to sort elements & it is avaialble in java.lang package.

We can sort the list of elements of comparable type by collection. Sort(List) method.

The comparator provides multiple sorting sequence we can sort element .