1. Write a java program to create an user defined exception called PayOutOfBoundsException. This exception is thrown when basicpay is not in between 10000 and 30000.

import java.util.\*;

class PayOutOfBoundsException extends Exception{

String s;

PayOutOfBoundsException(String s){

this.s=s;

}

public String toString(){

return (s);

}

}

public class Main{

public static void main(String[] args) {

int basicpay=20000;

try{

if(basicpay<10000 || basicpay >30000)

throw new PayOutOfBoundsException("PayOutOfBoundsException");

else

System.out.println("NoException");

}catch(PayOutOfBoundsException e){

System.out.println(e);

}

}

}

Output:

NoException

1. Write a java program to create two threads which display a message every half second.

import java.util.\*;

import java.lang.\*;

public class Main implements Runnable{

Thread t;

public void run() {

for (int i = 7; i < 13; i++) {

System.out.println(Thread.currentThread().getName() + " " + i);

try {

Thread.sleep(500);

} catch (Exception e) {

System.out.println(e);

}

}

}

public static void main(String[] args) {

Thread t = new Thread(new Main());

t.start();

Thread t2 = new Thread(new Main());

t2.start();

}

}

Output:

Thread-0 7

Thread-1 7

Thread-0 8

Thread-1 8

Thread-0 9

Thread-1 9

Thread-0 10

Thread-1 10

Thread-0 11

Thread-1 11

Thread-1 12

Thread-0 12

1. Write a java program to implement interthread communication.

import java.util.\*;

import java.lang.\*;

class Chat {

boolean flag = false;

public synchronized void Question(String msg) {

if (flag) {

try {

wait();

}catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(msg);

try {

Thread.sleep(1000);

} catch (Exception e) {}

flag = true;

notify();

}

public synchronized void Answer(String msg) {

if (!flag) {

try {

wait();

}catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(msg);

try {

Thread.sleep(1000);

} catch (Exception e) {}

flag = false;

notify();

}

}

class T1 implements Runnable {

Chat m;

String[] s1 = { "Hi", "How are you ?", "I am also doing fine!" };

public T1(Chat m1) {

this.m = m1;

new Thread(this, "Question").start();

}

public void run() {

for (int i = 0; i < s1.length; i++) {

m.Question(s1[i]);

}

}

}

class T2 implements Runnable {

Chat m;

String[] s2 = { "Hi", "I am good, what about you?", "Great!" };

public T2(Chat m2) {

this.m = m2;

new Thread(this, "Answer").start();

}

public void run() {

for (int i = 0; i < s2.length; i++) {

m.Answer(s2[i]);

}

}

}

public class Main {

public static void main(String[] args) {

Chat m = new Chat();

new T1(m);

new T2(m);

}

}

Output:

Hi

Hi

How are you ?

I am good, what about you?

I am also doing fine!

Great!

1. Write a java program to implement Thread Synchronization.

import java.util.\*;

import java.lang.\*;

class SendMessage{

public synchronized void send(String s){

System.out.println("Sending "+s);

try{

Thread.sleep(1000);

}catch(Exception e){}

System.out.println("message sent "+s);

}

}

class SenderThread extends Thread{

private String msg;

SendMessage sm;

SenderThread(String s,SendMessage m){

msg=s;

sm=m;

}

public void run(){

sm.send(msg);

}

}

public class Main {

public static void main(String[] args) {

SendMessage sm=new SendMessage();

SenderThread t1=new SenderThread("Hello",sm);

SenderThread t2=new SenderThread("Ajay",sm);

t1.start();

t2.start();

}

}

Output:

Sending Hello

message sent Hello

Sending Ajay

message sent Ajay

1. Write a java program to implement Generic Class,Generic Method and Generic Constructor.

import java.util.\*;

class G<T>{

T t;

G(T t){

this.t=t;

}

public T getT(){return t;}

public <E> void display(E element){

System.out.println(element);

}

}

public class Main {

public static void main(String[] args) {

G<String> g=new G<String>("Hello World");

System.out.println(g.getT());

g.display("Hello");

g.display(5);

}

}

Output:

Hello World

Hello

5

1. Write a java program to count no of vowels in a given file.

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String[] args) throws Exception{

File f=new File("E:\\rightstroke ass\\sample.txt");

BufferedReader br=new BufferedReader(new FileReader(f));

String v="aeiouAEIOU";

String text="";

int count=0;

String s;

while((s=br.readLine())!=null){

System.out.print(s);

text+=s;

}

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

for (int i=0;i<v.length() ;i++ ) {

if(text.charAt(j)==v.charAt(i)){

count+=1;

break;

}

}

System.out.println("\n"+count);

}

}

Output:

Hello World

3

1. Write a java program to implement autoboxing and unboxing.

import java.io.\*;

class Main

{

public static void main (String[] args)

{

Integer i = new Integer(5);

int i1 = i;

System.out.println("Value of i: " + i);

System.out.println("Value of i1: " + i1);

Character c = 'a';

char ch = c;

System.out.println("Value of ch: " + ch);

System.out.println("Value of c: " + c);

}

}

Output:

Value of i: 5

Value of i1: 5

Value of ch: a

Value of c: a

1. Write a java program to copy a file.

import java.io.\*;

class Main {

public static void main(String[] args) {

byte[] array = new byte[100];

try {

FileInputStream sourceFile = new FileInputStream("E:\\rightstroke ass\\sample.txt");

FileOutputStream destFile = new FileOutputStream("E:\\rightstroke ass\\ajay.txt");

sourceFile.read(array);

destFile.write(array);

System.out.println("The sample.txt file is copied to ajay.txt.");

sourceFile.close();

destFile.close();

}

catch (Exception e) {}

}

}

file copied

1. Write a java program to implement Stack using Generic class.

import java.io.\*;

class EmptyStackException extends RuntimeException {

public EmptyStackException() {

this("Stack is empty");

}

public EmptyStackException(String exception) {

super(exception);

}

}

class Stack<T>{

private int n;

private T[] stack;

private int top;

Stack(int n){

this.n=n;

top=-1;

stack=(T[]) new Object[n];

}

void push(T element){

if(top==n-1)

System.out.println("StackOverflowError");

else{

top+=1;

stack[top]=element;

}

}

<T> void pop(){

if(top==-1){

System.out.println("StackUnderflowException");

}else{

top-=1;

}

}

T peek(){

if(top!=-1)

return stack[top];

else

throw new EmptyStackException();

}

boolean isEmpty(){

return (top==-1);

}

boolean isFull(){

return(top==n-1);

}

int search(T element){

int pos=-1;

for(int i=0;i<top;i++)

if(element==stack[i]){

pos=i;

break;

}

return pos;

}

<T> void show(){

for(int i=0;i<=top;i++){

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

}

System.out.println();

}

}

class Main {

public static void main(String[] args) {

Stack<Integer> s=new Stack<>(5);

s.push(1);

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

s.push(2);

s.push(3);

s.push(4);

s.push(5);

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

s.show();

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

s.pop();

s.show();

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

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

}

}

Output:

1

5

1 2 3 4 5

true

1 2 3 4

false

false

1. Write java program to swap two values using generic method.

import java.io.\*;

class Main<T> {

T a,b,c;

public void swap(Main m){

m.c=m.a;

m.a=m.b;

m.b=m.c;

}

public static void main(String[] args) {

Main<Integer> m=new Main<>();

m.a=2;

m.b=4;

System.out.println("Before swap a="+m.a+",b="+m.b);

m.swap(m);

System.out.println("After swap a="+m.a+",b="+m.b);

}

}

Output:

Before swap a=2,b=4

After swap a=4,b=2

1. What is thread?

A thread is a:

* Facility to allow multiple activities within a single process
* Referred as lightweight process
* A thread is a series of executed statements
* Each thread has its own program counter, stack and local variables
* A thread is a nested sequence of method calls
* Its shares memory, files and per-process state

1. Write the difference between multithreading and multitasking

**Multitasking:**

Multitasking is when a CPU is provided to execute multiple tasks at a time. Multitasking involves often CPU switching between the tasks, so that users can collaborate with each program together. Unlike multithreading, In multitasking, the processes share separate memory and resources. As multitasking involves CPU switching between the tasks rapidly, So the little time is needed in order to switch from the one user to next.

**Multithreading:**

Multithreading is a system in which many threads are created from a process through which the computer power is increased. In multithreading, CPU is provided in order to execute many threads from a process at a time, and in multithreading, process creation is performed according to cost. Unlike multitasking, multithreading provides the same memory and resources to the processes for execution.

1. What is Enumeration?

**Enumeration** means a list of named constant. In Java, enumeration defines a class type. An Enumeration can have constructors, methods and instance variables. It is created using enum keyword. Each enumeration constant is public, static and final by default. Even though enumeration defines a class type and have constructors, you do not instantiate an enum using new. Enumeration variables are used and declared in much a same way as you do a primitive variable.

1. What is autoboxing?

**Autoboxing:** Converting a primitive value into an object of the corresponding wrapper class is called autoboxing. For example, converting int to Integer class.

.

1. What is wrapper class?

A **Wrapper class** is a class whose object wraps or contains a primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store a primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

1. what is transient modifier?

**transient** is a variables modifier used in serialization. At the time of serialization, if we don’t want to save value of a particular variable in a file, then we use transient keyword. When JVM comes across transient keyword, it ignores original value of the variable and save default value of that variable data type.

1. What is Generic class?Write the syntax of generic class.

A **generic class** declaration looks like a non-generic class declaration, except that the class name is followed by a type parameter section.

Syntax:

public class classname<T> {

……

}

1. What is stream?

 A **stream** is a sequence of objects that supports various methods which can be pipelined to produce the desired result

9.What is predefined stream?

**Predefined Streams**

Java provides three predefined stream objects: **in**, **out**, and **err**, defined in the System class of the java.lang package. The out object refers to the standard output stream or console. The in object refers to standard input, which is the keyboard. And, the err object refers to a standard error, which again is nothing but the console. As should be obvious, they may be redirected to any other compatible I/O devices, because System.in is nothing but an object of InputStream, and System.out and System.err are objects of the PrintStream class. So, they basically work on a byte-oriented stream although we can use them for reading and writing characters to and from the console.

10.What is multithreading?

**Multithreading** is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

1. What is the use of toString()?

The **toString** method is used to return a string representation of an object. If any object is printed, the toString() method is internally invoked by the java compiler. Else, the user implemented or overridden toString() method is called.

12.What is deadlock?

**Deadlock in java**

Deadlock in java is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, that is acquired by another thread and second thread is waiting for an object lock that is acquired by first thread. Since, both threads are waiting for each other to release the lock, the condition is called deadlock.

1. Write inter thread communication methods.

* wait()
* notify()
* notifyAll()

14.Write the difference between Checked and Unchecked exception.

difference between checked and unchecked exception is that the checked exceptions are checked at compile-time while unchecked exceptions are checked at runtime.

15. What is thread synchronization?

**Synchronization in Java**

Synchronization in java is the capability to control the access of multiple threads to any shared resource.

Java Synchronization is better option where we want to allow only one thread to access the shared resource.