Assignment-4

**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 UserException{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

int basicpay=sc.nextInt();

try{

if(basicpay<10000 || basicpay>30000)

throw new PayOutOfBoundsException();

}

catch(PayOutOfBoundsException e){

System.out.println(e) ;

}

}

}

class PayOutOfBoundsException extends Exception{

public String toString(){

return ("Invalid basicpay") ;

}

}

**Output:**

Invalid basicpay

**2.** **Write a java program to create two threads which display a message every half second?**

class thread1 extends Thread

{

thread1(String s)

{

super(s);

}

public void run()

{

for(int i=0;i<10;i++) {

String j = Thread.currentThread().getName();

System.out.println(j+" :- hello");

try

{

Thread.currentThread().sleep(500);

}

catch(InterruptedException e)

{

System.out.println(e);

}

}

}

}

class thread2 extends Thread

{

thread2(String s)

{

super(s);

}

public void run()

{

for(int i=0;i<10;i++) {

String j = Thread.currentThread().getName();

System.out.println(j+":- Everyone");

try

{

Thread.currentThread().sleep(500);

}

catch(InterruptedException e)

{

System.out.println(e);

}

}

}

}

class Mainthread

{

public static void main(String args[])

{

thread1 t1=new thread1("Thread 1");

thread2 t2=new thread2("Thread 2");

t1.start();

t2.start();

}

}

**Output:**

C:\programs>javac Mainthread.java

C:\programs>java Mainthread

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

Thread 1 :- hello

Thread 2:- Everyone

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

Thread 2:- Everyone

Thread 1 :- hello

**3. Write a java program to implement interthread communication?**

class Chat {

boolean flag = false;

public synchronized void Question(String msg) {

if (flag) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(msg);

flag = true;

notify();

}

public synchronized void Answer(String msg) {

if (!flag) {

try {

wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

System.out.println(msg);

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 TestThread {

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!

**4.Write a java program to implement Thread Synchronization.**

import java.lang.\*;

class trial extends Thread

{

synchronized void show(String s)

{

System.out.println("hello");

try

{

Thread.sleep(500);

}

catch(Exception e)

{

}

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

}

}

class A extends Thread

{

trial t;

A(trial t1){

t=t1;

}

public void run()

{

t.show("java");

}

}

class B extends Thread

{

trial t;

B(trial t1){

t=t1;

}

public void run()

{

t.show("c++");

}

}

class C extends Thread

{

trial t;

C(trial t1)

{

t=t1;

}

public void run()

{

t.show("c");

}

}

class test

{

public static void main(String[] args)

{

trial t2=new trial();

A a1=new A(t2);

B b1=new B(t2);

C c1=new C(t2);

a1.start();

b1.start();

c1.start();

}

}

**Output:**

Hello

javaworld

hello

cworld

hello

Terminated

Execution Timed Out!

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

import java.util.\*;

class Generic<T>

{

T obj;

<T1> Generic (T1 ob1) {

obj = (T)ob1;

}

public <T2> void getObject(T2 e) {

System.out.println(obj+" "+e);

}

}

class Demo

{

public static void main (String[] args)

{

Generic <Integer> iObj = new Generic <Integer>(20);

iObj.getObject("srilakshmi");

Generic <String> sObj =

new Test Generic <String>("String");

sObj.getObject(srilu);

}

}

**Output:**

20 srilakshmi

String srilu

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

import java.io.\*;

public class Count {

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

Count in = null;

try {

in = new Count("vowels.txt");

int i,c=0;

char ch;

while ((i = in.read()) != -1) {

ch=(char)i;

if(ch=='a' || ch=='e' || ch=='i' || ch=='o' || ch=='u' || ch=='A' || ch=='E' || ch=='I' || ch=='O' || ch=='U' )

c+=1;

}

System.out.print("Number of vowels : "+c);

in.close();

}

catch(Exception e){

System.out.println(e);

}

}

}

Vowels.txt:

My name is srilakshmi

**Output:**

Number of vowels : 6

**7.Write a java program to implement autoboxing and unboxing**.

class Demo

{

public static void main (String[] args)

{

Integer i = new Integer(44);

int i1 = i;

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

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

char ch = ‘a’;

Character ch1= new Character(ch);

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

System.out.println("Value of bo: " +ch1);

}

}

**Output:**

Value of i:44

Value of i1:44

Value of ch:a

Value of bo:a

**8.Write a java program to copy a file.**

public class CopyExample

{

public static void main(String[] args)

{

FileInputStream instream = null;

FileOutputStream outstream = null;

try{

File infile =new File("E:\\vowels.txt");

File outfile =new File("E:\\vowels1.txt");

instream = new FileInputStream(infile);

outstream = new FileOutputStream(outfile);

byte[] buffer = new byte[1024];

int length;

while ((length = instream.read(buffer)) > 0){

outstream.write(buffer, 0, length);

}

instream.close();

outstream.close();

System.out.println("File copied successfully!!");

}catch(IOException ioe){

ioe.printStackTrace();

}

}

}

**Output:**

E:\>javac CopyExample.java

E:\>java CopyExample

File copied successfully!!

**9.Write a java program to implement Stack using Generic class**

import java.util.\*;

public class GenericStack <T> {

private ArrayList<T> stack = new ArrayList<T> ();

private int top = 0;

public int size () { return top; }

public void push (T item) {

stack.add (top++, item);

}

public T pop () {

return stack.remove (--top);

}

public static void main (String[] args) {

GenericStack<Integer> s = new GenericStack<Integer> ();

s.push (4);

int s1=s.size();

System.out.println("size of stack: "+s1);

int i = s.pop ();

System.out.println("top element: "+i);

}

}

**Output:**

size of stack: 1

top element: 4

**10.Write java program to swap two values using generic method.**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class Main {

public static final <T> void swap (T[] a, int i, int j) {

T temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

private static void test() {

String [] a = {"sri", "lakshmi"};

System.out.println("original array : "+Arrays.toString(a));

swap(a, 0, 1);

System.out.println("Swap using array : "+Arrays.toString(a));

}

public static void main(String args[])

{

test();

}

}

**Output:**

Original array: [sri, lakshmi]

Swap using array : [lakshmi,sri]

**1.What is thread?**

A thread is an independent path of execution within a program. Many threads can run concurrently within a program. Every thread in Java is created and controlled by the java.lang.Thread class. A Java program can have many threads, and these threads can run concurrently, either asynchronously or synchronously.

**2. Write the difference between multithreading and multitasking?**

* In multitasking, several programs are executed concurrently e.g. Java compiler and a Java IDE like [Netbeans](http://java67.blogspot.sg/2013/02/how-to-connect-mysql-database-from-java.html) or [Eclipse](http://java67.blogspot.com/2014/04/how-to-make-executable-jar-file-in-Java-Eclipse.html), while in multi-threading multiple threads execute either same or different part of program multiple times at the same time.
* Multi-threading is more granular than multi-tasking. In multi-tasking,  CPU switches between multiple programs to complete their execution in real time, while in multi-threading CPU switches between multiple threads of the same program. Remember, switching between multiple processes has more context switching cost than switching between multiple threads of the same program.
* Process are heavyweight as compared to threads, they require their own address space, which means multi-tasking is heavy compared to multithreading. Inter-process communication is expensive and limited and context switching from one process to another is expensive and limited.

**3.** **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.

**4. What is autoboxing?**

Converting a primitive value into an object of the corresponding [wrapper class](https://www.geeksforgeeks.org/wrapper-classes-java/) is called autoboxing. For example, converting int to [Integer class](https://www.geeksforgeeks.org/wrapper-classes-java/). The Java compiler applies autoboxing when a primitive value is:

* Passed as a parameter to a method that **expects an object** of the corresponding wrapper class.
* Assigned to a variable of the corresponding **wrapper class.**

**5. 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.

**6. What is transient modifier?**

**Transient** is a variables modifier used in [serialization](http://quiz.geeksforgeeks.org/serialization-in-java/). 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.

**7. 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.

As with generic methods, the type parameter section of a generic class can have one or more type parameters separated by commas. These classes are known as parameterized classes or parameterized types because they accept one or more parameters.

**Creating Genric class:**

**class** MyGen<T>{

T obj;

**void** add(T obj){**this**.obj=obj;}

T get(){**return** obj;}

}

**8.** **What is stream?**

 A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.  
The features of Java stream are –

* A stream is not a data structure instead it takes input from the Collections, Arrays or I/O channels.
* Streams don’t change the original data structure, they only provide the result as per the pipelined methods.
* Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

**9. What is predefined stream?**

Every language has some predefined streams for its users and Java is one of these languages. Three Java Predefined streams or standard streams are available in the java.lang.System class. These are as follows:

|  |  |
| --- | --- |
| **System.in** | This is the standard stream for **input**. This stream is used for reading data for the program from the keyboard by default. |
| **System.out** | This is the standard stream for **output**. This stream is used for writing data from the program to an output device such as a monitor / **console** by default or to some specified file. |
| **System.err** | This is a standard stream for **error**. This is used to show an error message on the screen i.e. **console** by default for the users. |

**10. What is multithreading?**

**Multithreading in**[Java](https://www.javatpoint.com/java-tutorial) is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

**11.What is the use of toString()?**

**Use of toString():**

* The first implementation is when it is called as a method of an object instance. The example below shows this implementation
* The second implementation is when you call the member method of the relevant class by passing the value as an argument. The example below shows how to do this

**12.What is deadlock?**

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.

**13.** **Write inter thread communication methods?**

Interthread communication enables a more subtler control it. Although event loop programming is a gross alternative to multithreading, the benefit of using a thread to realize multitasking far outweighs event loops. The event loop leads to polling. Polling waits for a certain event to occur by repeatedly checking for a certain condition. This means that the consumer of the event or the event handler has to be recurrent just about the production of event, without doing any actual processing. This is clearly a waste of CPU time. And, once the event is generated, polling starts, which further wastes CPU time.

**14. Write the difference between Checked and Unchecked exception?**

**Checked:** are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword. **Unchecked** are the exceptions that are not checked at compiled time. In C++, all exceptions are unchecked, so it is not forced by the compiler to either handle or specify the exception. It is up to the programmers to be civilized, and specify or catch the exceptions.

**15. What is thread synchronization?**

Thread synchronization is the concurrent execution of two or more threads that share critical resources. Threads should be synchronized to avoid critical resource use conflicts. Otherwise, conflicts may arise when parallel-running threads attempt to modify a common variable at the same time.