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

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

**class** PayOutOfBoundsException **extends** Exception{

**private** **int** pay;

**public** PayOutOfBoundsException(**int** pay) {

**this**.pay=pay;

}

**public** String toString(){

**return** "Invalid Pay";

}

}

**public** **class** pgm1 {

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

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

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

**try**{

**if**(pay<=10000 || pay>=30000){

**throw** **new** PayOutOfBoundsException(pay);

}

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

}

**catch**(PayOutOfBoundsException e){

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

}

}

}

Output:

12

Inavlid Pay

10001

10001

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

**class** Thread1 **extends** Thread{

**public** **void** run(){

**try** {

**for**(**int** i=0;i<5;i++){

System.***out***.println("Thread1");

Thread.*sleep*(500);

}

} **catch** (InterruptedException e) {

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

}

}

}

**class** Thread2 **extends** Thread{

**public** **void** run(){

**try** {

**for**(**int** i=0;i<5;i++){

System.***out***.println("Thread2");

Thread.*sleep*(500);

}

} **catch** (InterruptedException e) {

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

}

}

}

**public** **class** pgm2 {

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

Thread1 t1=**new** Thread1();

t1.start();

Thread2 t2=**new** Thread2();

t2.start();

}

}

Output:

Thread1

Thread2

Thread1

Thread2

Thread2

Thread1

Thread2

Thread1

Thread1

Thread2

3.Write a java program to implement interthread communication.

**class** IntThread{

**private** **int** amount=1000;

**synchronized** **void** withdraw(**int** amount){

**if**(**this**.amount<amount){

**try**{

System.***out***.println("waiting for deposit..");

wait();

System.***out***.println("Amount withdrawn.");

}

**catch**(Exception e){

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

}

}

}

**synchronized** **void** deposit(**int** amount){

**this**.amount+=amount;

System.***out***.println("Amount deposited.");

notify();

}

}

**public** **class** pgm3 {

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

**final** IntThread p=**new** IntThread();

**new** Thread(){

**public** **void** run(){

p.withdraw(1500);

}

}.start();

**new** Thread(){

**public** **void** run(){

p.deposit(1000);

}

}.start();

}

}

Output:

waiting for deposit..

Amount deposited.

Amount withdrawn.

4.Write a java program to implement Thread Synchronisation.

**class** Table{

**synchronized** **public** **void** printTable(**int** n){//synchronized method

**for**(**int** i=1;i<=5;i++){

System.***out***.println(n\*i);

**try**{

Thread.*sleep*(400);

}**catch**(Exception e){System.***out***.println(e);}

}

}

}

**class** Thread11 **extends** Thread{

Table t;

Thread11(Table t){

**this**.t=t;

}

**public** **void** run(){

t.printTable(5);

}

}

**class** Thread22 **extends** Thread{

Table t;

Thread22(Table t){

**this**.t=t;

}

**public** **void** run(){

t.printTable(100);

}

}

**public** **class** pgm4 {

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

Table obj=**new** Table();

Thread11 t1=**new** Thread11(obj);

Thread22 t2=**new** Thread22(obj);

t1.start();

t2.start();

}

}

Output:

5

10

15

20

25

100

200

300

400

500

5.Write a java program to implement Generic class,Generic method and Generic interface.

**public** **class** pgm5<T> {

**private** T t;

**public** <E> **void** print(E[] arr){

**for**(E a:arr){

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

}

}

**public** **void** add(T t){

**this**.t=t;

}

**public** T get(){

**return** t;

}

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

pgm5<Integer> p=**new** pgm5<Integer>();

pgm5<String> q=**new** pgm5<String>();

q.add(**new** String("Hello"));

p.add(**new** Integer(10));

Integer[] intarr={1,2,3,4};

String[] strarr={"A","B","C","D"};

p.print(intarr);

q.print(strarr);

System.***out***.println(p.get()+" "+q.get());

}

}

Output:

1

2

3

4

A

B

C

D

10 Hello

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

**package** File;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.IOException;

**public** **class** pgm6

{

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

{

File f1=**new** File("input.txt");

String[] words=**null**;

FileReader fr = **new** FileReader(f1);

BufferedReader br = **new** BufferedReader(fr);

String s;

**int** count=0;

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

{

words=s.split(" ");

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

{

**for**(**int** j=0;j<words[i].length();j++)

{

**char** ch=words[i].charAt(j);

**if**(ch == 'a' || ch == 'e' || ch == 'i' ||ch == 'o' || ch == 'u') //Checking for vowels

{

count+=1;

}

}

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

}

}

}

}

Output:

23

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

**public** **class** pgm6{

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

**int** a=50;

Integer a2=5;//Boxing

**int** a3=a2;

System.***out***.println(a2+" "+a3);

}

}

Output:

5 5

8.Write a java program to copy a file.

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**class** Main {

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

**byte**[] array = **new** **byte**[50];

**try** {

FileInputStream sourceFile = **new** FileInputStream("input.txt");

FileOutputStream destFile = **new** FileOutputStream("newFile");

// reads all data from input.txt

sourceFile.read(array);

// writes all data to newFile

destFile.write(array);

System.***out***.println("The input.txt file is copied to newFile.");

// closes the stream

sourceFile.close();

destFile.close();

}

**catch** (Exception e) {

e.getStackTrace();

}

}

}

Output:

The input.txt file is copied to newFile.

9.Write a java program to implement stack using generic class,

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

**public** **class** pgm6 <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) {

pgm6<Integer> s = **new** pgm6<Integer> ();

s.push (17);

**int** i = s.pop ();

System.***out***.format ("%d%n", i);

}

}

Output:

17

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

**class** Pair<T> {

T first;

T second;

}

**public** **class** pgm6 <T> {

**public** **static** <T> **void** swap(Pair<T> p) {

T temp = p.first;

p.first = p.second;

p.second = temp;

}

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

Pair<Integer> p=**new** Pair<>();

p.first=2;

p.second=3;

*swap*(p);

System.***out***.println(p.first+" "+p.second);

}

}

Output:

3 2

/---------------------Theoretical Questions-------------------------/

1.What is thread?

Ans:

A thread is a small part of a program that executes concurrently with the other parts of the program.

2.Difference between multithreading and multitasking.

Ans:

In **multitasking**, several programs are executed concurrently e.g. **Java** compiler and a **Java** IDE like Netbeans or Eclipse, while in **multi-threading multiple threads** execute either same or different part of program multiple times at the same time.

3.What is Enumeration?

Ans:

**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.

4.What is autoboxing?

Ans:

**Autoboxing** is the automatic conversion that the **Java** compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called unboxing.

5.What is wrapper class?

Ans:

**Wrapper classes** provide a way to use primitive data types ( int , boolean , etc..) as objects.

6.What is transient modifier?

Ans:

**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.

7.What is Generic class?write the syntax of generic class.

Ans:

A class that operates on a parameterized type is called a generic class.

Ex:

**class** Pair<T> {

T first;

T second;

}

8.What is a stream?

Ans:

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?

Ans:

**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. ... out and System.

10.What is multithreading?

Ans:

**multiple threads** execute either same or different part of program multiple times at the same time.

11.What is the use of toString()?

Ans:

The **toString** method is used to return a string representation of an object

12.What is deadlock?

Ans:

**Deadlock** describes a situation where two or more threads are blocked forever, waiting for each other.

13.Write interthread communication methods?

Ans:

1.wait()

2.notify()

3.notifyAll()

14.Write the difference between checked and unchecked exception.

Ans:

Exceptions that are checked at the compile time are called checked exception where exceptions checked at the run time are called unchecked exceptions.

15.What is thread synchronization?

Ans:

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