**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.**

**Source Code :**

**import** java.util.Scanner;

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

**public** **class** UserDefExcep {

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

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

System.***out***.print("Enter the Amount to be paid : ");

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

sc.close();

**try** {

**if**(basicpay>=10000 && basicpay <= 30000)

System.***out***.print("Thank you for selecting the correct amount ! ");

**else**

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

}

**catch**(PayOutOfBoundsException e) {

**if**(basicpay<10000)

System.***out***.println("Your Amount is less than 10,000!");

**else**

System.***out***.println("Your Amount is greater than 30,000!");

}

}

}

**Output :**







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

**Source Code :**

**public** **class** MyThread {

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

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

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

t1.start();

t2.start();

}

}

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

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

**try** {

System.***out***.print("Thread 1 Started \n");

**for**(**int** i=8;i>0;i--) {

Thread.*sleep*(500);

System.***out***.println("T1 : "+i);

}

}

**catch** (InterruptedException e) {

e.printStackTrace();

}

**finally** {

System.***out***.println("Thread 1 Completed");

}

}

}

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

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

**try** {

System.***out***.println("Thread 2 Started ");

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

Thread.*sleep*(500);

System.***out***.println("T2 : "+i);

}

}

**catch** (InterruptedException e) {

e.printStackTrace();

}

**finally** {

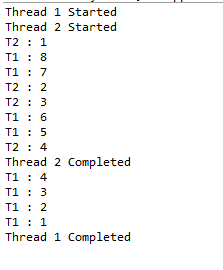
System.***out***.println("Thread 2 Completed");

}

}

}

**Output :**



1. **Write a java program to implement interthread communication.**

**Source Code :**

**import** java.util.Scanner;

**class** Customer{

**int** amount =10000;

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

System.***out***.println("Going to withdraw.... Amount is : "+amount);

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

System.***out***.println("Less balance; Waiting to deposit ...");

**try** {

wait();

}

**catch**(Exception e) { }

}

**else** {

**this**.amount-=amount;

System.***out***.println("Withdraw Completed.....\nReamining Balance is : "+**this**.amount);

}

}

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

System.***out***.println("---Another Thread---\nGoing to deposit...");

**this**.amount+=amount;

System.***out***.println("Deposit completed... \n Total Balance is : "+**this**.amount);

notify();

}

}

**public** **class** InterProComm {

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

**final** Customer c =**new** Customer();

System.***out***.println("Amount is : 10000 ");

**new** Thread(){

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

c.withdraw(10000);

}

}.start();

**new** Thread(){

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

c.deposit(15000);

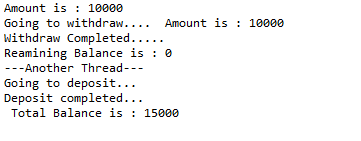
}

}.start();

}

}

**Output :**



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

**Source Code :**

**class** Table{

**synchronized** **void** printTable(**int** n) {

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

System.***out***.println(n+" \* "+i+" = "+(n\*i));

**try** {

Thread.*sleep*(400);

}

**catch**(Exception e) {

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

}

}

}

**class** MyThread1 **extends** Thread{

Table t;

MyThread1(Table t){

**this**.t=t;

}

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

t.printTable(5);

}

}

**class** MyThread2 **extends** Thread{

Table t;

MyThread2(Table t){

**this**.t=t;

}

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

t.printTable(100);

}

}

**class** Sync{

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

Table t =**new** Table();

MyThread1 t1 = **new** MyThread1(t);

MyThread2 t2 = **new** MyThread2(t);

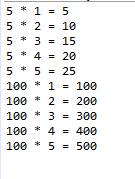
t1.start();

t2.start();

}

}

**Output :**



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

**Source Code :**

**class** Gen<T>{//generic class

T ob;

Gen(T o){//generic constructor

ob=o;

}

**public** T getOb() {

**return** ob;

}

}

**public** **class** GenericExample {

**public** **static** <E> **void** printArray(E[] inputArray) {//generic block

**for**(E element:inputArray) {

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

}

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

}

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

Gen <Integer> iob = **new** Gen<>(100);

**int** x=iob.getOb();

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

Gen <String> sob = **new** Gen<>("Hellooo");

String s=sob.getOb();

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

Integer[] intArray = {1,2,3,4,5};

Double[] doubleArray= {5.8,4.5,2.3,3.3,8.6};

Character[] charArray = {'S','A','T','Y','A'};

System.***out***.print("Array integer contains : ");*printArray*(intArray);

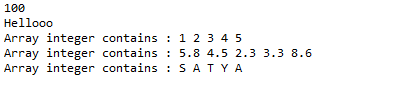
System.***out***.print("Array integer contains : ");*printArray*(doubleArray);

System.***out***.print("Array integer contains : ");*printArray*(charArray);

}

}

**Output :**



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

**Source Code :**

**import** java.util.Scanner;

**import** java.io.FileNotFoundException;

**import** java.io.File;

**public** **class** CountVowels{

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

**int** vCount = 0;

File file = **new** File("Sample.txt");

Scanner sc = **new** Scanner(file);

sc.useDelimiter("\\Z");

**while**(sc.hasNextLine()){

String str=sc.nextLine();

str = str.toLowerCase();

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

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

vCount++;

}

}

}

System.***out***.println("Number of vowels: " + vCount);

}

}

**Output :**



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

**Source Code :**

**public** **class** AutoUnBox {

**static** **int** add(Integer a, Integer b) {//autoboxing in parameters

**return** a+b;//unboxing in return

}

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

**int** sum=*add*(10,22);

System.***out***.println("Sum = "+sum);

}

}

**Output :**



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

**Source Code :**

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** FileCopy

{

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

{

File fsrc=**new** File("Sample.txt"); //File Descriptor for source file

File fdes=**new** File("output.txt"); //File Descriptor for destination file

FileReader fr=**new** FileReader(fsrc); //Creation of file reader object

BufferedReader br=**new** BufferedReader(fr); //Creation of buffer reader object

FileWriter fw= **new** FileWriter(fdes); //Creation of file writer object

String s=**null**;

**while**((s=br.readLine())!=**null**) //Copying Content to the new file

{

fw.write(s);

fw.write("\n");

fw.flush();

}

fw.close();

System.***out***.print("Please Check the File \" Output.txt \" ");

}

}

**Output :**



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

**Source Code :**

**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>();

GenericStack<String> st =**new** GenericStack<String>();

s.push(17);

st.push("Satya");

s.push(33);

s.push(3);

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

String str=st.pop();

System.***out***.format("Popped from stack is : %d",i);

System.***out***.format("\nPopped from stack is : "+str);;

}

}

**Output :**



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

**Source Code :**

**public** **class** GenericSwap {

**static** <T> **void** swap(T first, T second) {

System.***out***.println(" Before Swapping ");

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

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

T temp = first;

first=second;

second=temp;

System.***out***.println("After Swapping ");

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

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

}

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

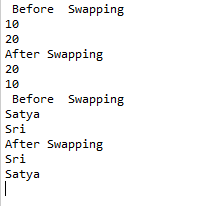
*swap*(10,20);

*swap*("Satya","Sri");

}

}

**Output :**



**1. What is thread?**

A thread is an independent path of execution within a program. A java program can have many threads, and these threads can concurrently, either asynchronously or synchronously.

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

|  |  |
| --- | --- |
| **Multithreading** | **Multitasking** |
| 1.In multithreading, many threads are created from a process through which computer power is increased. | 1. In multitasking, users are allowed to perform many tasks by CPU. |
| 2.CPU switching is often involved between the threads. | 2.CPU switching between the tasks. |
| 3.In multithreading, processes are allocated same memory. | 3.While in multitasking, the processes share separate memory. |
| 4.Component does not involve multiprocessing. | 4.Component involves multiprocessing. |
| 5.Each process share same resources. | 5.Processes don't share same resources, each process is allocated separate recources. |
| 6.Termination of thread takes less time. | 6.Termination of process takes more time. |

**3. What is Enumeration?**

It is one of the predefined interface present in java.util.\* package.

The purpose of enumeration interface object is that to extract the data from any legacy collection framework variable in forward direction only.

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?**

Autoboxing refers to the automatic conversion of a primitive type variable to its corresponding wrapper class object. The compiler automatically handles the conversion when a primitive value is passed as an argument to a function which a wrapper class object. assigned to a variable of the type wrapper class.

**5. What is wrapper class?**

The wrapper class in java provides the mechanism to convert the primitives into object and object into primitive.

**6. what is transient modifier?**

The transient modifier tells the java object serialization subsystem to exclude the field when serializing an instance of the class. When the object is then deserialized, the field will be initialized to the default value; i.e., Null for a reference type, and zero or false for a primitive type.

**7. What is Generic class?Write the syntax of generic class.**

A class is generic if it declares one or more type variables. These type variables are known as the type parameters of the class.

**Syntax : Example:**

public class Generic\_class<Type>{ public class ArrayList<E>{

} }

**8. What is stream?**

A stream is an abstraction that either produces or consumes information. Java provides two types of input stream and output stream.

* The InputStream is used to read data from a source
* The OutputStream is used for writing data to a destination.

java.io package is implements all the stream classes.

**9.What is predefined stream?**

Java Provides three predefined stream objects : in ,out, and err defined in the System class of the java.lang packages.

* The out object refers to the standared output stream or console.
* The in object refers to standard input, which is the keyboard.
* The err object refers to a standard error.

**10.What is multithreading?**

Multithreading is a process that allows execution of two or more threads simultaneously to maximum utilization of CPU. A thread is a lightweight sub-process, the smallest unit of processing.

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

A toString() is an in-built method in java that returns the value given to it String format. any object that this method is applied on , will then be returned as a string object. The toString() method is used to return a string representation of an object.

**12.What is deadlock?**

Deadlock is a part of multithreading. Deadlock can occur in a situation when a thread is waiting for an object lock, i.e., acquired by another thread and second thread is waiting for an object lock that is acquires 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.**

The process of exchanging the data/information between multiple threads with the consistent common is known as Inter Thread Communication.

Inter Thread Communication Methods present in java.lang.object. They are :

1. public final void wait(long)
2. public final void wait()
3. public final void notify()
4. public final void notifyAll()

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

|  |  |
| --- | --- |
| **Checked Exception** | **Unchecked Exception** |
| 1.Checked Exceptions occur at compile time. | 1.Unchecked Exceptions occur at runtime. |
| 2.The compiler checks a checked exception. | 2.The compiler does not check these types of exceptions. |
| 3.These types of exceptions can be handled at the time of compilation. | 3.These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program. |
| 4.They are the sub-class of the exception class. | 4.They are runtime exceptions and hence are not a part of the Exception class. |
| 5.JVM needs the exception to catch and handle. | 5.JVM does not require the exception to catch and handle. |
| 6.Examples of checked exceptions :   * File Not Found Exception * No Such File Exception * No Such Method Exception * Class Not Found Exception * Interrupted Exception | 6.Examples of unchecked exceptions :   * Security Exception * Array Index Out Of Bounds Exeption * No Such Element Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Undeclared Throwable Exception |

**15. What is thread synchronization?**

The process of allowing one thread among multiple threads into the shareable area for perform write/ read operations for getting result by eliminating inconsistent results is called synchronization