**Student ID : IT17076326**

**Name : Hemachandra M. A. J. C.**

**Session : 1**

**Generics :**

public class Box<T> **{ /\***for String,

private T t; Scanner scn = new Scanner(System.in);

public void add(T t) { String name = scn.next();\*/

this.t = t;}

public T get() {

return t;} \*Static methods can be called

public static void main(String[] args) { directly using the class name.

Box<Integer> integerBox = new Box<Integer>(); [Classname].[static method(with balue)]

Box<String> stringBox = new Box<String>();

integerBox.add(new Integer(10)); Create Array in genarics:

stringBox.add(new String("Hello World")); int marks[] = new int[5];

System.out.printf("Integer Value :%d\n\n", integerBox.get());

System.out.printf("String Value :%s\n", stringBox.get());

} Collection<Integer> value = new ArrayList<>();

**}** value.add(5);

**Singleton**

class Login **{ /**public static <E> void printArray(E[] elements){

private static Login myLogin; for(E elem; elements){

private Login() { } System.out.print(elem + “ ”);

public static Login getInstance() { }System.out.println();}**/**

if (myLogin == null) {

myLogin = new Login(); T-Type; E-Element; K-Key; N-Number; V-Value;

} \*public class A <T extends Number>{}

return myLogin;

}

boolean validateUser(String user, String password) {

if (password.equals(user))

return true;

else

return false;

}

**}**

**Exception handling**

public class InsufficientBalanceException extends Exception **{**

private double amount;

public InsufficientBalanceException(double amount) {

this.amount = amount;

}

public double getAmount() {

return amount;

}

**}**

try {

continueTransaction(account);

} catch (InsufficientBalanceException e) {

System.out.println("Sorry, your account remains only Rs." + e.getAmount());

e.printStackTrace();

} finally {

System.out.println("Do you wish to continue? yes/no");

Scanner scanner = new Scanner(System.in);

if(scanner.next().equalsIgnoreCase("yes")){

try {

System.out.println("Depositing Rs.10,000");

account.deposit(10000.00);

continueTransaction(account);

} catch (InsufficientBalanceException e) {

e.printStackTrace();

}

}else

System.out.println("Good Bye");

public static void continueTransaction(Account account) throws InsufficientBalanceException {

System.out.print("Please enter amount to be withdrawn = ");

Scanner scanner = new Scanner(System.in);

double amount;

while (true) {

amount = scanner.nextDouble();

System.out.print("\nWithdrawing Rs." + amount + "/=");

account.withdraw(amount);

System.out.println(" existing amount = " + account.getBalance());

System.out.print("Please enter amount to be withdrawn = ");

class PrintThread extends Thread { //wait for main thread die

public void run() { object.join();

for (int r = 1; r <= 100; r++) { //tempory stop main thread & execute new thread

System.out.println(r); object.yield();

}}} //in thread1

class SLIITThread implements Runnable { public void run(){

public void run() { synchronized(object){

for (int r = 1; r <= 100; r++) { try{

System.out.println(" SLIIT "); object.wait();

} }} }}}

class Ques1a {

public static void main(String args[]) { //in thread2

PrintThread thread1 = new PrintThread(); public void run(){

Thread thread2 = new Thread(new SLIITThread()); synchronized(object){

thread1.start(); object.notify();

thread2.start();

} }

class NumbersThread extends Thread {

public NumbersThread(){}

public void run() {

synchronized(this){

for (int r = 1; r <= 100; r++) {

System.out.println("Thread Name : " + Thread.currentThread().getName() + " - " + r);

}}}}

class Ques1b {

public static void main(String args[]) {

NumbersThread thread1 = new NumbersThread();

thread1.setName("Red");

for (int r = 1; r <= 100; r++){

System.out.println("SLIIT");

}

thread1.start();

System.out.println("Thread 1 - State = " + thread1.getState() + " - Alive = " + thread1.isAlive());

}}