**Problem 1**

**package** problem1;

**import** java.util.Scanner;

**public** **class** Solution1 {

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

System.***out***.print("Please write a positive numnber or zero in range from 0 to 100 ");

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

String ch = sc.next();

**if**(!ch.matches("^[\\-d|\\d]+$"))

{

**throw** **new** IllegalArgumentException("Only allow numeric values");

}

Integer value=Integer.*parseInt*(ch);

**if**(value>100)

{

**throw** **new** IllegalArgumentException("Your value is more greater that 100 ");

}

**if**(value<0)

{

**throw** **new** IllegalArgumentException(" Upps your value is les that zero");

}

**else**

{

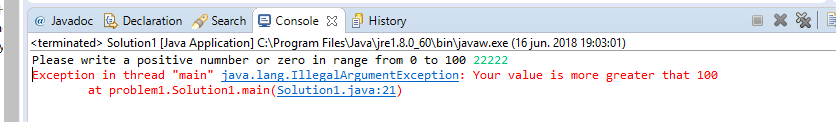
System.***out***.printf("Upa your value is correct %s \n" ,value );

**return** ;

}

}

}



Problem 2

**package** problem2;

**import** java.util.Scanner;

**public** **class** CustomerAccount {

String Cus\_name ;

String Acc\_No ;

Double Balance ;

**private** **final** Double MAX\_VALUE\_ON\_ACCCOUNT=100.0;

**public** CustomerAccount(String cus\_name, String acc\_No, Double balance) {

**if** (balance<**this**.MAX\_VALUE\_ON\_ACCCOUNT )

{

**throw** **new** CustomerAccountException(String.*format*("Withdraw :Balance Initial reach below 100$ : to new customer %s intial is %s Check and try again ", cus\_name,balance));

}

**this**.Cus\_name = cus\_name;

**this**.Acc\_No = acc\_No;

**this**.Balance = balance;

}

@Override

**public** String toString() {

**return** String.*format*("|Name(%s) :<Id:%s, Balance:$%,.2f>|", **this**.Cus\_name,**this**.Acc\_No,**this**.Balance);

}

**public** **boolean** Withdraw(Double value)

{

**if** (value<=0)

{

**throw** **new** CustomerAccountException("Withdraw : only positive values ​​are allowed");

}

**if** (value > **this**.Balance)

{

**throw** **new** CustomerAccountException(String.*format*("Withdraw (Withdraw amount exceed the balance):Amount not allowed : can not Witdraw %s because your balance is %s", value,**this**.Balance) );

}

**if** (**this**.Balance-value>=**this**.MAX\_VALUE\_ON\_ACCCOUNT )

{

**throw** **new** CustomerAccountException(String.*format*("Withdraw :Balance reach below 100$ : request is %s and your current balance %s", value,**this**.Balance));

}

**this**.Balance=**this**.Balance-value;

**return** **true**;

}

**public** **boolean** deposit(Double value)

{

**if** (value<=0)

{

**throw** **new** CustomerAccountException(String.*format*("Deposit : only positive values ​​are allowed , value <%s>", value) );

}

**this**.Balance=**this**.Balance+value;

**return** **true**;

}

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

CustomerAccount pedro= **new** CustomerAccount("Pedro", "A32241E", 170.0);

pedro.deposit(130.0);

**try**

{

pedro.deposit(320.0);

}

**catch** (CustomerAccountException e)

{

System.***err***.println(e.getMessage()+"\n");

}

**try**

{

pedro.deposit(-92.0);

}

**catch** (CustomerAccountException e)

{

System.***err***.println(e.getMessage()+"\n");

}

**try**

{

pedro.Withdraw(700.0);

}

**catch** (CustomerAccountException e)

{

System.***err***.println(e.getMessage()+"\n");

}

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

**try**

{

**new** CustomerAccount("Farid", "A232241E", 10.0);

}

**catch** (CustomerAccountException e)

{

System.***err***.println(e.getMessage()+"\n");

}

}

}

**package** problem2;

**public** **class** CustomerAccountException **extends** IllegalArgumentException {

**public** CustomerAccountException(String s)

{

**super**(s);

}

}

