# Question 1

Lab Exercise on Exception Handling

Write an application that throws and catches an ArithmeticException when you attempt to take the square root of a negative value. Prompt the user for an input value and try the Math.sqrt() method on it. The application either displays the square root or catches the thrown Exception and displays an appropriate message. Save the file as SqrtException.java.

# CODE:

import java.util.Scanner; public class SqrtException

{

public static void main(String[] args)

{

Scanner in = new Scanner (System.in); System.out.print("Enter a number: "); double n= in.nextInt();

try

{

if(n<0)

throw new ArithmeticException("Negative value is not allowed for square root calculation"); else

System.out.println("Square root of "+n+": "+Math.sqrt(n));

}

catch(ArithmeticException a)

{

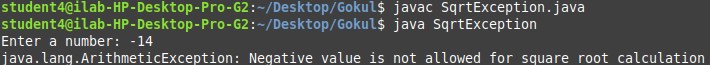
System.out.println(a);

}

}

}

# OUTPUT:



**Question 2**

The Double.parseDouble() method requires a String argument, but it fails if the String cannot be converted to a floating-point number. Write an application in which you try accepting a double input from a user and catch a NumberFormatException if one is thrown. The catch block forces the number to 0 and displays an appropriate error message. Following the catch block, display the number. Save the file as TryToParseDouble.java.

# CODE:

import java.util.Scanner; public class TryToParseDouble

{

public static void main(String[] args)

{

Scanner in = new Scanner (System.in); System.out.print("Enter a number: "); String s = in.next();

Double value; try

{

value=Double.parseDouble(s);

}

catch(NumberFormatException obj)

{

System.out.println(obj+" cannot be converted into a number"); value=0.0;

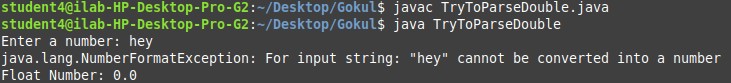
}

System.out.println("Float Number: "+value);

}

}

# OUTPUT:



**Question 3**

Define Employee class with Employee code, name, date of birth and date of appointment. The

Employee code must have the format of year-designation-number. The year is a two digit integer

such as 87. the designation is a single letter code M for manager, A for Administrative staff, H for

HR dept staff, E for Executive staff, and T for Technical staff. The number is a three digit number.

The following are some sample employee codes.

82-M-183

76-A-242

71-H-107

Write a Java program to read the employee code, name, date of birth, and date of appointment and

validate the employee code. If the employee code is incorrect a suitable user defined exception must

be thrown. Then verify if date of birth is before date of appointment. If it is not so, then throw

another user defined Exception. If it is correct, then create the Employee object, display the count of

employee and display the details of employees.

**CODE:**

import java.util.Scanner;

class InvalidECodeException extends Exception

{

InvalidECodeException(String msg)

{

super(msg);

}

}

class InvalidDOAException extends Exception

{

InvalidDOAException(String msg)

{

super(msg);

}

}

class Employee

{

static String ecode, name, dob, doa;

Employee(String ecode, String name, String dob, String doa)

{

this.ecode=ecode;

this.name=name;

this.dob=dob;

this.doa=doa;

}

}

public class TestEmployee

{

public static boolean isNumber(String str)

{

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

{

if(str.charAt(i) >= '0' && str.charAt(i) <= '9')

return true;

}

return false;

}

public static boolean isAlpha(char c)

{

if(c=='M' || c=='A' || c=='H' || c=='E' || c=='T')

return true;

else

return false;

}

public static boolean isValid(String ecode) throws InvalidECodeException

{

if (ecode.length()==8)

{

if(isNumber(ecode.substring(0,2)) && isAlpha(ecode.charAt(3)) && isNumber(ecode.substring(5,8)))

return true;

else

throw new InvalidECodeException("Invalid employee code");

}

else

throw new InvalidECodeException("Invalid employee code");

}

public static boolean isBefore(String dob, String doa) throws InvalidDOAException

{

if (Integer.valueOf(dob.substring(6,10))<Integer.valueOf(doa.substring(6,10)))

return true;

else if(Integer.valueOf(dob.substring(6,10))>Integer.valueOf(doa.substring(6,10)))

throw new InvalidDOAException("DOB is not before DOA");

else

{

if(Integer.valueOf(dob.substring(3,5))<Integer.valueOf(doa.substring(3,5)))

return true;

else if(Integer.valueOf(dob.substring(3,5))>Integer.valueOf(doa.substring(3,5)))

throw new InvalidDOAException("DOB is not before DOA");

else

{

if(Integer.valueOf(dob.substring(0,2))<Integer.valueOf(doa.substring(0,2)))

return true;

else

throw new InvalidDOAException("DOB is not before DOA");

}

}

}

public static void main(String[] args)

{

String ecode, name, dob, doa;

Scanner in = new Scanner(System.in);

int count=0, i;

System.out.print("Enter no:of employees: ");

int n=in.nextInt();

Employee[] e= new Employee[n];

for(i=0;i<n;i++)

{

System.out.print("\nEnter ecode: ");

ecode=in.next();

System.out.print("Enter name : ");

name=in.next();

System.out.print("Enter dob : ");

dob=in.next();

System.out.print("Enter doa : ");

doa=in.next();

try

{

if(isValid(ecode) && isBefore(dob,doa))

{

e[count]=new Employee(ecode, name, dob, doa);

count++;

}

}

catch (InvalidECodeException e1)

{

System.out.println("Exception occured: " + e1);

}

catch (InvalidDOAException e2)

{

System.out.println("Exception occured: " + e2);

}

}

System.out.println("\nNo: of valid Employees: "+count);

for(i=0;i<count;i++)

{

System.out.println("\nEmployee-"+(i+1)+": ");

System.out.println("Ecode: "+e[i].ecode);

System.out.println("Name : "+e[i].name);

System.out.println("DOB : "+e[i].dob);

System.out.println("DOA : "+e[i].doa);

}

}

}

**OUTPUT:**

