1)Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class, implement a constructor that uses both father and son’s age and throws an exception if son’s age is >=father’s age.

A)

```
class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message);
    }
}

class Father {
    int age;

    public Father(int age) throws WrongAge {
        if (age < 0) {
            throw new WrongAge("Father's age cannot be negative");
        }
        this.age = age;
    }
}

class Son extends Father {
    int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAge {
        super(fatherAge);
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be equal or greater than the father's
age");
        }
        this.sonAge = sonAge;
    }
}

public class Main {
    public static void main(String[] args) {
```

```
        try {
            Son son = new Son(40, 42);
        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

Output:
Exception: Son's age cannot be equal or greater than the father's age

2)Design a program where students do the course registration, Input is USN, Name, Courses etc.
Generate the User defined exceptions if wrong USN or Name or semester is given.

A)

```
class InvalidUSNException extends Exception {
    public InvalidUSNException(String message) {
        super(message);
    }
}

class InvalidNameException extends Exception {
    public InvalidNameException(String message) {
        super(message);
    }
}

class InvalidSemesterException extends Exception {
    public InvalidSemesterException(String message) {
        super(message);
    }
}

class Student {
    String usn, name;
    int semester;
```

```
   public Student(String usn, String name, int semester) throws InvalidUSNException,
InvalidNameException, InvalidSemesterException {
       if (!usn.matches("[1-9][A-Z]{2}[0-9]{4}")) {
           throw new InvalidUSNException("Invalid USN format");
       }
       if (name.isEmpty()) {
           throw new InvalidNameException("Name cannot be empty");
       }
       if (semester < 1 || semester > 8) {
           throw new InvalidSemesterException("Invalid semester");
       }
       this.usn = usn;
       this.name = name;
       this.semester = semester;
   }
}

public class Main {
   public static void main(String[] args) {
       try {
           Student student = new Student("1ABC2023", "", 5);
       } catch (Exception e) {
           System.out.println("Exception: " + e.getMessage());
       }
   }
}
```

Output:
Enter USN: 1WA23CS015
Enter Name: Tarun S
Enter Semester: 9
Error: Semester must be between 1 and 8
Course Registration Completed.

4) Solve the problem given in the hacker link below
https://www.hackerearth.com/problem/algorithm/exception-handling-2-2711f586-7d0a57bc/

A)

```
import java.util.Scanner;
```

```java
public class Solution {

   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);
       int n = scanner.nextInt();
       int[] arr = new int[n];
       for (int i = 0; i < n; i++) {
           arr[i] = scanner.nextInt();
       }
       String str = scanner.next();
       try {
           int result = 10 / 0;
           System.out.println("a");
       } catch (ArithmeticException e) {
           System.out.println("Invalid division");
       }
       try {
           int num = Integer.parseInt(str);
           System.out.println("a");
       } catch (NumberFormatException e) {
           System.out.println("Format mismatch");
       }
       try {
           char c = str.charAt(str.length());
           System.out.println("a");
       } catch (StringIndexOutOfBoundsException e) {
           System.out.println("Index is invalid");
       }
       try {
           int element = arr[arr.length];
           System.out.println("a");
       } catch (ArrayIndexOutOfBoundsException e) {
           System.out.println("Array index is invalid");
       }
       try {
           throw new MyException(117);
       } catch (MyException e) {
           System.out.println(e.getMessage());
```

```
        } catch (Exception e) {
            System.out.println("Exception encountered");
        } finally {
            System.out.println("Exception Handling Completed");
        }
    }
}

class MyException extends Exception {
    private int param;
    public MyException(int param) {
        this.param = param;
    }

    @Override
    public String getMessage() {
        return "MyException[" + param + "]";
    }
}
```

Output:
10
10
9
8
7
6
5
4
3
2
1
0
Invalid division
a
Index is invalid
Array index is invalid
MyException[117]
Exception Handling Completed

3) Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box.

A)

```
import java.util.Scanner;

public class DivisionCalculator {
   public static void main(String[] args) {
       Scanner scanner = new Scanner(System.in);

       while (true) {
           try {
               System.out.print("Enter num1: ");
               int num1 = Integer.parseInt(scanner.nextLine());

               System.out.print("Enter the num2: ");
               int num2 = Integer.parseInt(scanner.nextLine());

               if (num2 == 0) {
                   throw new ArithmeticException("Cannot divide by zero.");
               }

               int result = num1 / num2;
               System.out.println("Result: " + result);
           } catch (NumberFormatException ex) {
               System.out.println("Error: Please enter valid integers.");
           } catch (ArithmeticException ex) {
               System.out.println("Error: " + ex.getMessage());
           }

           System.out.print("Want to continue? (yes/no): ");
           String choice = scanner.nextLine();
           if (!choice.equalsIgnoreCase("yes")) {
               break;
           }
       }
```

```
        System.out.println("The end");
        scanner.close();
    }
}
```

Output:
Enter num1: 10
Enter num2: 2
Result: 5
Want to continue? (yes/no): yes

Enter num1: 15
Enter num2: 0
Error: Cannot divide by zero.
Want to continue? (yes/no): no

The end

5)Complete the examples of Threads discussed in the class.

A)

```
class NewThread implements Runnable {
   Thread t;
  String s;
   NewThread(String s1) {
  s=s1;
      t = new Thread(this, s);
      System.out.println("Child thread: " + t);
      t.start(); // Start the thread
   }
public void run() {
       try {
         for(int i = 5; i > 0; i--) {
              System.out.println("Child Thread: " + i+"\t"+s);
                Thread.sleep(500);
          }
      } catch (InterruptedException e) {
```

```
            System.out.println("Child interrupted.");
    }
        System.out.println("Exiting child thread.");
   }
}

class ThreadDemo {
 public static void main(String args[]) {
      NewThread t1 = new NewThread("A1");
      NewThread t2 = new NewThread("B2");
      NewThread t3 = new NewThread("C3");
      NewThread t4 = new NewThread("D4");
      NewThread t5 = new NewThread("E5");
      NewThread t6 = new NewThread("F6");
      NewThread t7 = new NewThread("G7");
      NewThread t8 = new NewThread("H8");
      NewThread t9 = new NewThread("I9");
      NewThread t10 = new NewThread("J10");



      try {
          for(int i = 5; i > 0; i--) {
              System.out.println("Main Thread: " + i);
              Thread.sleep(1000);
          }
      } catch (InterruptedException e) {
     System.out.println("Main thread interrupted.");
      }
      System.out.println("Main thread exiting.");
    }
}
```

Output:
Child thread: Thread[A1,5,main]
Child thread: Thread[B2,5,main]
Child thread: Thread[C3,5,main]
Child thread: Thread[D4,5,main]
Child thread: Thread[E5,5,main]
Child thread: Thread[F6,5,main]

```
Child thread: Thread[G7,5,main]
Child thread: Thread[H8,5,main]
Child thread: Thread[I9,5,main]
Child thread: Thread[J10,5,main]
Main Thread: 5
Child Thread: 5	A1
Child Thread: 5	B2
Child Thread: 5	C3
Child Thread: 5	D4
Child Thread: 5	E5
Child Thread: 5	F6
Child Thread: 5	G7
Child Thread: 5	H8
Child Thread: 5	I9
Child Thread: 5	J10
Main Thread: 4
Child Thread: 4	A1
Child Thread: 4	B2
Child Thread: 4	C3
Child Thread: 4	D4
Child Thread: 4	E5
Child Thread: 4	F6
Child Thread: 4	G7
Child Thread: 4	H8
Child Thread: 4	I9
Child Thread: 4	J10
Main Thread: 3
Child Thread: 3	A1
Child Thread: 3	B2
Child Thread: 3	C3
Child Thread: 3	D4
Child Thread: 3	E5
Child Thread: 3	F6
Child Thread: 3	G7
Child Thread: 3	H8
Child Thread: 3	I9
Child Thread: 3	J10
Main Thread: 2
Child Thread: 2	A1
Child Thread: 2	B2
Child Thread: 2	C3
Child Thread: 2	D4
Child Thread: 2	E5
```

Child Thread: 2 F6
Child Thread: 2 G7
Child Thread: 2 H8
Child Thread: 2 I9
Child Thread: 2 J10
Main Thread: 1
Child Thread: 1 A1
Child Thread: 1 B2
Child Thread: 1 C3
Child Thread: 1 D4
Child Thread: 1 E5
Child Thread: 1 F6
Child Thread: 1 G7
Child Thread: 1 H8
Child Thread: 1 I9
Child Thread: 1 J10
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Exiting child thread.
Main thread exiting.

6) Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.
A)

```
public class MultiThreadedMessages {
   public static void main(String[] args) {
       Thread thread1 = new Thread(new Runnable() {
           @Override
           public void run() {
               try {
                   while (true) {
                       System.out.println("BMS College of Engineering");
                       Thread.sleep(10000);
                   }
               } catch (InterruptedException e) {
```

```
                    System.out.println("Thread 1 break");
                }
            }
        });

        Thread thread2 = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    while (true) {
                        System.out.println("CSE");
                        Thread.sleep(2000);
                    }
                } catch (InterruptedException e) {
                    System.out.println("Thread 2 break");
                }
            }
        });

        thread1.start();
        thread2.start();
    }
}
```

Output:
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
...