WEEK 8 :

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.

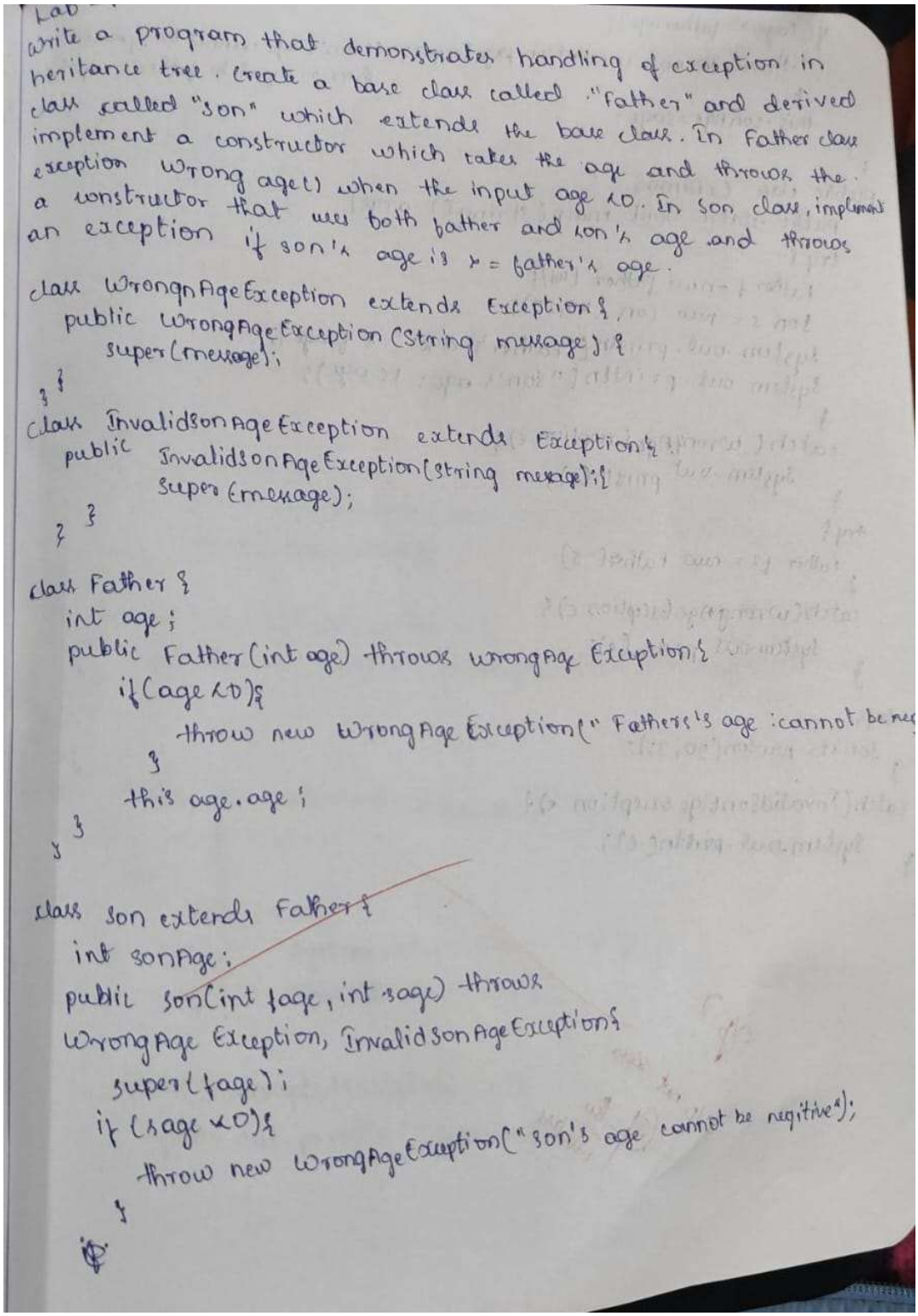
Source Code :

```
class ThreadDemo extends Thread{
    public void run(){
    while(true){
    System.out.println("BMS College Of Engineering");
    try{
    Thread.sleep(10000);
    }
    catch(InterruptedException e){
    e.printStackTrace();
    }
     }
  }
}

class CSEThread extends Thread{
    public void run(){
    while(true){
    System.out.println("CSE");
    try{
    Thread.sleep(2000);
    }
    catch(InterruptedException e){
    e.printStackTrace();
    }
     }
  }
}

public class Demo{
    public static void main(String[] args){
    ThreadDemo t1 = new ThreadDemo();
    CSEThread t2 = new CSEThread();
    t1.start();
    t2.start();
     }
 }
```

Output :

```
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
```

Written code and output:



Lab

Write a program that demonstrates handling of exception in heritance 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.

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

class Father {
    int age;
    public Father(int age) throws WrongAgeException {
        if(age<0){
            throw new WrongAgeException("Father's age :cannot be neg
        }
        this.age.age;
    }
}

class Son extends Father {
    int sonAge;
    public Son(int fage, int sage) throws
    WrongAgeException, InvalidSonAgeException{
        super(fage);
        if(sage<0){
            throw new WrongAgeException("Son's age cannot be negitive");
        }
    }
```

```
    if(sage >= fatherage){
      throw new InvalidSonAgeException("Son's age cannot be greater
                                        than or equal to father's age");
    }
    this.sonAge = sage;
  }
}
public class ExDemo5{
  public static void main(String[] args)
  try{
    Father f = new Father(40);
    Son s = new Son(40,20);
    System.out.println("Father's age:" + f.age);
    System.out.println("Son's age: " + s.age);
  }
  catch(WrongAgeException e){
    System.out.println(e);
  }
  try{
    Father f2 = new Father(-5)
  }
  catch(WrongAgeException e){
    System.out.println(e);
  }
try{
  Son s2 = new Son(30,35);
}
catch(InvalidSonAgeException e){
  System.out.println(e);
```

o/p ?