Practical 3

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/1		
Aim:	WAP to read the given text file.Convert contents to lowercase and store		

Source code:

```
import java.io.*;


class file_read{
   public static void main(String[] args){
      BufferedReader br=null;
      BufferedWriter bw=null;
      try{
         br =new BufferedReader(new FileReader("TestFile1.txt"));
         bw=new BufferedWriter(new FileWriter("LowercaseTestFile1.txt", true));


         String line =br.readLine();


         while(line != null){
             bw.write(line.toLowerCase()+ "\n");
             line = br.readLine();
         }
         bw.flush();
         System.out.println("reading-writting complete!!");
      }catch(IOException e){
         System.out.println("unable to read file"+ e);
      }
      finally{
            if(br!=null)
```

```
            try{
              br.close();
            }catch(Exception e){System.out.println("file not opended");}
        }
    }
}
```

Output:

reading-writting complete!!

TestFile1.txt	**LowercaseTestFile1.txt**
Hi,I am Shiv	hi,i am shiv
How Are You	how are you

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/2		
Aim:	WAP to read the given text file.Search for particular word and replace that word		

Source code:

```
import java.io.*;


class file_read_2{
   public static void main(String[] args){
      BufferedReader br=null;
      BufferedWriter bw=null;
      try{
         br =new BufferedReader(new FileReader("TestFile2.txt"));
         bw=new BufferedWriter(new FileWriter("replaceFile2.txt", true));
         /*to read file  */
         String line =br.readLine();
```

```
    while(line != null){
        bw.write(line.replace("very", "not"));
        line = br.readLine();
    }
    bw.flush();
    System.out.println("reading-writting complete!!");
}catch(IOException e){
    System.out.println("unable to read file"+ e);
}
finally{
        if(br!=null)
        try{
          br.close();
        }catch(Exception e){System.out.println("file not opended");}
    } } }
```

Output:

reading-writting complete!!

TestFile2.txt	**replaceFile2.txt**
Hi,shiv is very good in maths.	Hi,shiv is not good in maths.
he is very clever.	he is not clever.

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/3		
Aim:	WAP to remove duplicate lines from the given source file.and store it to another file		

Source code:

```
import java.io.*;
import java.util.HashSet;
```

```
class RemoveDuplicateLines {
    public static void main(String[] args) {
        BufferedReader br = null;
        BufferedWriter bw = null;
        try {
            br = new BufferedReader(new FileReader("source.txt"));
            bw = new BufferedWriter(new FileWriter("target.txt"));

            String line;
            HashSet<String> uniqueLines = new HashSet<>();

            while ((line = br.readLine()) != null) {
                if (uniqueLines.add(line)) {

                    bw.write(line);
                    bw.newLine(); // Add a newline to separate lines in the target file
                }
            }
            bw.flush();
            System.out.println("Duplicate lines removed and stored in target file.");
        } catch (IOException e) {
            System.out.println("Unable to read/write file: " + e);
        } finally {
            try {
                if (br != null)
                    br.close();
                if (bw != null)
                    bw.close();
            } catch (IOException e) {
```

```
            System.out.println("Error while closing the file: " + e);
        } } } }
```

Output:

Duplicate lines removed and stored in target file.

source.txt	**target.txt**
Hi,I am shiv	Hi,I am shiv
Hi,I am shiv	
Hi,I am shiv	

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/4		
Aim:	WAP to store the subject mark and student name to the file "markdeails.txt" .Read the file "markdetails.txt" and print the name of students having marks more than 70.		

Source code:

```
import java.io.*;
public class Main {
   public static void main(String[] args) {
      BufferedReader br = null;
      BufferedWriter bw = null;
      try {
         br = new BufferedReader(new FileReader("markdetails.txt"));
         String line = br.readLine();
         System.out.println("Students with marks more than 70:");
         while (line != null) {
            // Split the line into student name and marks
            String[] parts = line.split(" ");
```

```
        String name = parts[0];
        int marks = Integer.parseInt(parts[1]);

        if (marks > 70) {
            System.out.println(name);
        }
        line = br.readLine();
    }
    System.out.println("Reading and complete!");
}
catch (IOException e) {
    System.out.println("Unable to read or write files: " + e);
}
finally {
    try {
        if (br != null) {
            br.close();
        }
        if (bw != null) {
            bw.close();
        }
    }
    catch (IOException e) {
        System.out.println("Error while closing the files.");

    } } } }
```

Output: Students with marks more than 70:

shiv

jagrat

jishan

vivek

akshat

kartik

Reading complete!

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/5		
Aim:	WAP to store Book object details(bookTitle, bookPrice, bookId) to the file bookdetails.dat. bookdetails.dat".(minimum 5 records).Read the "bookdetails.dat, search for a book having particular title, print the booktile having highest price.		

Source code:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Collections;

class MyBook implements Serializable {
  private String title;
  private double price;
  private int id;

  public MyBook(String title, double price, int id) {
    this.title = title;
    this.price = price;
    this.id = id;
  }

  public String getTitle() {
    return title;
  }

  public double getPrice() {
    return price;
  }

  public int getId() {
    return id;
  }
```

```
  @Override
  public String toString() {
    return "Book [Title=" + title + ", Price=" + price + ", ID=" + id + "]";
  }
}

public class MyBookStore {
  public static void main(String[] args) {
    List<MyBook> myBooks = new ArrayList<>();

    myBooks.add(new MyBook("The Alchemist", 15.99, 101));
    myBooks.add(new MyBook("To Kill a Mockingbird", 12.49, 202));
    myBooks.add(new MyBook("1984", 10.75, 303));
    myBooks.add(new MyBook("Pride and Prejudice", 13.99, 404));
    myBooks.add(new MyBook("The Great Gatsby", 14.50, 505));

    try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream("mybookdetails.dat"))) {
      for (MyBook book : myBooks) {
        oos.writeObject(book);
      }
    } catch (IOException e) {
      e.printStackTrace();
    }

    String titleToSearch = "1984";
    MyBook foundMyBook = null;

    try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream("mybookdetails.dat"))) {
      while (true) {
        MyBook book = (MyBook) ois.readObject();
        if (book.getTitle().equals(titleToSearch)) {
          foundMyBook = book;
          break;
        }
      }
    } catch (EOFException e) {
      // End of file reached
    } catch (IOException | ClassNotFoundException e) {
      e.printStackTrace();
    }

    if (foundMyBook != null) {
      System.out.println("Found my book with title: " + foundMyBook.getTitle());
    } else {
      System.out.println("My book not found.");   }
```

```
    MyBook myBookWithHighestPrice = Collections.max(myBooks, (b1, b2) ->
        Double.compare(b1.getPrice(), b2.getPrice()));
    System.out.println("My book with the highest price: " + myBookWithHighestPrice);
  } }
```

Output:

Found my book with title: 1984

My book with the highest price: Book [Title=The Alchemist, Price=15.99, ID=101]

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/6		
Aim:	WAP to create Threads		

Source code:

```
 import java.util.*;
class child_thread1 extends Thread{
   public void run(){
   int i=0;
   for(i=0;i<=250;i++){
      System.out.println("thread1, value:"+i);
   } } }
class child_thread2 implements Runnable{
   public void run(){
  int i=250;
  for(i=250;i<=500;i++){
   System.out.println("thread2, value:"+i);
  }  }   }
public class practical3_6{
   public static void main(String[] args){
      child_thread1 thread1=new child_thread1();
      thread1.start();
      child_thread2 thread2=new child_thread2();
```

```
    Thread thread = new Thread(thread2);


    thread.start();
    thread1.setPriority(4);
    thread.setPriority(2);


   for (int i=501;i<700;i++){
    System.out.println("main thread hello, value:"+i);
   }  }  }
```

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/7		
Aim:	childThread1 : prints the multiplication table of n1, while printing,it sleeps for 0.5 second.childThread2 : prints the multiplication table of n2. ChildThread2 must wait for thread1 to complete.mainThread : prints the multiplication table of n3.Ensure that main thread must wait for thread1 and thread2 to complete.		

Source code:

```
class ChildThread1 extends Thread{
 public void run(){
  for(int i=1;i<11;i++){
    System.out.println("childThread1,  2 x "+ i +"= "+2*i);
  }
  try {
          Thread.sleep(500); // Sleep for 0.5 seconds
       } catch (InterruptedException e) {
          e.printStackTrace();  }
 }
}

class ChildThread2 extends Thread{
  public void run(){
  for(int i=1;i<11;i++){
    System.out.println("childThread2,  3 x "+ i +"= "+2*i);
  }
  }
}

public class practical3_7 {
```

```
  public static void main(String[] args){
    ChildThread1 thread1 = new ChildThread1();
    ChildThread2 thread2 = new ChildThread2();

    thread1.start();

    try {
      thread1.join();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

    thread2.start();

    try {
      thread2.join();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }

    for(int i=1;i<11;i++){
    System.out.println("main-thread,  4 x "+ i +"= "+2*i);
  }
}
```

Output:

```
childThread1,  2 x 1= 2
childThread1,  2 x 2= 4
childThread1,  2 x 3= 6
childThread1,  2 x 4= 8
childThread1,  2 x 5= 10
childThread1,  2 x 6= 12
childThread1,  2 x 7= 14
childThread1,  2 x 8= 16
childThread1,  2 x 9= 18
childThread1,  2 x 10= 20
childThread2,  3 x 1= 3
childThread2,  3 x 2= 6
childThread2,  3 x 3= 9
childThread2,  3 x 4= 12
childThread2,  3 x 5= 15
childThread2,  3 x 6= 18
childThread2,  3 x 7= 21
childThread2,  3 x 8= 24
```

childThread2, 3 x 9= 27
childThread2, 3 x 10= 30
main-thread, 4 x 1= 4
main-thread, 4 x 2= 8
main-thread, 4 x 3= 12
main-thread, 4 x 4= 16
main-thread, 4 x 5= 20
main-thread, 4 x 6= 24
main-thread, 4 x 7= 28
main-thread, 4 x 8= 32
main-thread, 4 x 9= 36
main-thread, 4 x 10= 40

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/8		
Aim:	WAP to create a JointAccount banking application.Which allows withdraw and deposit functionality.When JointAccountHolders simultaneously trying to withdraw or deposit amount.Balance must be consistent.(Hint : Use synchronized). Demonstrate this scenario in main method.		

Source code:

```
class JointAccount {
  private double balance;

  public JointAccount(double initialBalance) {
    this.balance = initialBalance;
  }


 public void deposit( String Holder, double amount) {
    synchronized (this) {
      System.out.println(Holder + " is depositing :" + amount);
      balance += amount;
      System.out.println("New balance after deposit:" + balance);
    }
  }

  public void withdraw(String Holder,double amount) {
    synchronized (this) {
      if (balance >= amount) {
        System.out.println(Holder + " is withdrawing:" + amount);
        balance -= amount;
        System.out.println("New balance after withdrawing :" + balance);
```

```
        } else {
            System.out.println(Holder + " tried to withdraw " + amount + " but there's not enough
balance.");
        }
      }
    }

    public double getBalance() {
      return balance;
    }
}

public class prctical3_8 {
    public static void main(String[] args) {
      JointAccount account = new JointAccount(899.0);

      Thread accountHolder1 = new Thread(() -> {
        account.deposit("shiv",320.0);
        account.withdraw( "shiv",250.0);
      });

      Thread accountHolder2 = new Thread(() -> {
        account.deposit( "Akhil",560.0);
        account.withdraw("Akhil",550.0);
      });

      accountHolder1.start();
      accountHolder2.start();

      try {
        accountHolder1.join();
        accountHolder2.join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }

      System.out.println("Final balance in the joint account: " + account.getBalance());
    }
}
```

Output:

shiv is depositing :320.0
New balance after deposit:1219.0
shiv is withdrawing:250.0
New balance after withdrawing :969.0
Akhil is depositing :560.0

New balance after deposit:1529.0
Akhil is withdrawing:550.0
New balance after withdrawing :979.0
Final balance in the joint account: 979.0

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/9		
Aim:	WAP for producer consumer problem.		

Source code:

```
class Buffer {
   private int[] buffer;
   private int capacity;
   private int size;
   private int in;
   private int out;
   public Buffer(int capacity) {
      this.capacity = capacity;
      this.buffer = new int[capacity];
      this.size = 0;
      this.in = 0;
      this.out = 0;
   }

   public synchronized void produce(int item) {
      while (size == capacity) {
         try {
            wait();
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }

      buffer[in] = item;
      in = (in + 1) % capacity;
      size++;

      System.out.println("Produced: " + item);
      notify();
   }

   public synchronized int consume() {
      while (size == 0) {
         try {
            wait();
```

```
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        int item = buffer[out];
        out = (out + 1) % capacity;
        size--;

        System.out.println("Consumed: " + item);
        notify();
        return item;
    }
}

class Producer extends Thread {
    private Buffer buffer;

    public Producer(Buffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        for (int i = 1; i <= 10; i++) {
            buffer.produce(i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class Consumer extends Thread {
    private Buffer buffer;

    public Consumer(Buffer buffer) {
        this.buffer = buffer;
    }

    public void run() {
        for (int i = 1; i <= 10; i++) {
            int item = buffer.consume();
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
```

```
        }
      }
    }
}

public class practical3_9 {
    public static void main(String[] args) {
        Buffer buffer = new Buffer(5);

        Producer producer = new Producer(buffer);
        Consumer consumer = new Consumer(buffer);

        producer.start();
        consumer.start();
    }
}
```

Output:

```
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
Produced: 8
Consumed: 8
Produced: 9
Consumed: 9
Produced: 10
Consumed: 10
```

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/10		
Aim:	The Odd-Even Number Printing problem. It involves two threads,		

	one printing odd numbers and the other printing even numbers, alternatively.WAP to achieve this using wait() and notify().Output must be in sequence. Odd 1, Even 2, Odd 3, Even 4..upto 20.

Source code:

```
class OddEven{
  private int max;
  private int number = 1;
  private boolean isOdd = true;

  public OddEven(int max) {
    this.max = max;
  }

  public synchronized void printOdd() {
    while (number <= max) {
      while (!isOdd) {
        try {
          wait();
        } catch (InterruptedException e) {
          Thread.currentThread().interrupt();
        }
      }
      System.out.println("Odd: " + number);
      number++;
      isOdd = false;
      notify();
    }
  }

  public synchronized void printEven() {
    while (number <= max) {
      while (isOdd) {
        try {
          wait();
        } catch (InterruptedException e) {
          Thread.currentThread().interrupt();
        }
      }
      System.out.println("Even: " + number);
      number++;
      isOdd = true;
      notify();
    }
  }
}
```

```
public class practical3_10 {
  public static void main(String[] args) {
    OddEven printer = new OddEven(20);

    Thread oddThread = new Thread(() -> {
      printer.printOdd();
    });

    Thread evenThread = new Thread(() -> {
      printer.printEven();
    });

    oddThread.start();
    evenThread.start();

    try {
      oddThread.join();
      evenThread.join();
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }
}
```

Output:

```
Odd: 1
Even: 2
Odd: 3
Even: 4
Odd: 5
Even: 6
Odd: 7
Even: 8
Odd: 9
Even: 10
Odd: 11
Even: 12
Odd: 13
Even: 14
Odd: 15
Even: 16
Odd: 17
Even: 18
Odd: 19
```

Even: 20

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/11		
Aim:	WAP to create a user defined InvalidBoxException as an Unchecked Exception.Exception must be thrown from the constructor of the Box class if either of the dimension from length, width or height is zero or less than zero. Demonstrate in main.		

Source code:

```
class InvalidBoxException extends RuntimeException {

  public InvalidBoxException(String message) {
    super(message);
  }
}

class Box {

  private final double width;
  private final double height;
  private final double length;

  public Box(double length, double height, double width) {
    validateDimension(length);
    validateDimension(height);
    validateDimension(width);

    this.length = length;
    this.width = width;
    this.height = height;
  }

  private void validateDimension(double dimension) {
    if (dimension <= 0) {
      throw new InvalidBoxException("Dimension must be greater than zero.");
    }
  }

  public double calculateVolume() {
    return width * height * length;
  }
}

public class practical3_12 {
```

```
  public static void main(String[] args) {
    try {
      Box invalidBox = new Box(0.0, -1.0, 5.25);
      System.out.println("Invalid Box Volume: " + invalidBox.calculateVolume());
    } catch (InvalidBoxException e) {
      System.out.println("Error: " + e.getMessage());
    }

    Box validBox = new Box(2.0, 1.0, 1.0);
    System.out.println("Valid Box Volume: " + validBox.calculateVolume());
  }
}
```

Output:

Error: Dimension must be greater than zero.

Valid Box Volume: 2.0

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/13		
Aim:	WAP to create a user defined checked exception invalidAgeException will be thrown when applying for Vehiclelicense if age is less than 18. Demonstrate in main		

Source code:

```
class InvalidAgeException extends Exception {

  public InvalidAgeException(String message) {
    super(message);
  }
}

class LicenseApplication {
  public void applyForLicense(int age) throws InvalidAgeException {
    if (age < 18) {
      throw new InvalidAgeException("Your age must be grater or equal to 18 for application of
license");
    } else {
      System.out.println("License application approved for age " + age);
    }
  }
}
```

```
public class practical3_13 {

   public static void main(String[] args) {
      LicenseApplication licenseApp = new LicenseApplication();
      int applicantAge = 19;

      try {
         licenseApp.applyForLicense(applicantAge);
      } catch (InvalidAgeException e) {
         System.out.println("License application rejected: " + e.getMessage());
      }
   }
}
```

Output:

License application approved for age 19

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/14		
Aim:	WAP to demonstrate multiple catch block. WAP to show the use of nested try statements that emphasizes the sequence of checking for catch handler statements.		

Source code:

```
public class practical3_14 {
   public static void main(String[] args) {
      try {
         int[] numbers = { 1, 2, 3 };
         System.out.println(numbers[5]);
      } catch (ArrayIndexOutOfBoundsException e) {
         System.out.println("Caught ArrayIndexOutOfBoundsException: " + e.getMessage());
      } catch (NullPointerException e) {
         System.out.println("Caught NullPointerException: " + e.getMessage());
      } catch (ArithmeticException e) {
         System.out.println("Caught ArithmeticException: " + e.getMessage());
      }

      try {
         int result = divideByZero();
         System.out.println("Result: " + result);
      } catch (ArithmeticException e) {
```

```
      System.out.println("Caught ArithmeticException: " + e.getMessage());
    } catch (Exception e) {
      System.out.println("Caught Exception: " + e.getMessage());
    }
  }

  public static int divideByZero() {
    try {
      int numerator = 10;
      int denominator = 0;
      return numerator / denominator;
    } catch (ArithmeticException e) {
      System.out.println("Caught ArithmeticException in divideByZero: " + e.getMessage());
      throw e;
    }
  }
}
```

Output:

Caught ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3

Caught ArithmeticException in divideByZero: / by zero

Caught ArithmeticException: / by zero

ROLL NO:	22BCP317	Batch	G8
NAME:	Patel shiv vijaykumar		
Practical	3/15		
Aim:	WAP to handle ArrayIndexOutofBoundsException in java for binarySearch method.		

Source code:

```
public class practical3_15 {
  public static int binarySearch(int[] arr, int target) {
    int left = 0;
    int right = arr.length - 1;
    while (left <= right) {
      int mid = left + (right - left) / 2;

      if (arr[mid] == target) {
        return mid;
      }

      if (arr[mid] < target) {
        left = mid + 1;
```

```
        } else {
            right = mid - 1;
        }
    }
    return -1;
  }

  public static void main(String[] args) {
    int[ ] sortedArray = {1, 2, 3, 4, 5, 6, 7, 8, 9};

    try {
        int targetIndex = binarySearch(sortedArray, 5);
        if (targetIndex != -1) {
            System.out.println("Element found at index " + targetIndex);
        } else {
            System.out.println("Element not found in the array.");
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("ArrayIndexOutOfBoundsException occurred: " + e.getMessage());
    }
  }
}
```

Output:

Element found at index 4