Week 1

a) Write a java program that prints all real solutions to the quadratic equation ax2 +bx+c=0. Read in a, b, c and use the quadratic formula.

```
import java.util.Scanner;
class Equation
{
   public static void main (String args[])
   {
      System.out.println("Enter the cofficients a,b,c of quadratic equation");
      Scanner sc = new Scanner(System.in);
      double a=sc.nextInt();
        double b=sc.nextInt();
      double c=sc.nextInt();
      double z=b*b-4*a*c;
      if (z<0)
      {
         System.out.println("There are no real solutions");
      }
      else if(z==0)
      {
           double r1= (-b)/(2*a);
           System.out.println("The solutions are real and equal"+r1);
      }
      else
      {
         double r2= ((-b)+z)/(2*a);
         double r3= ((-b)-z)/(2*a);
           System.out.println("There exiists two real solutions"+r2+" and "+r3);

      }
    }
}
```

b) The Fibonacci sequence is defined by the following rule. The first two values in the sequence are 1 and 1. Every subsequent value is the sum of the two values preceding it. Write a java program that uses both recursive and non-recursive functions.

Non-recursive:

```
import java.util.Scanner;
class Fib{
public static void main(String args[])
{
System.out.println("Enter a number");
```

```
Scanner sc = new Scanner(System.in);
double a=sc.nextInt();
 int f=0,g=1;
 for(int i=0;i<a;i++)
 {
  System.out.print(f);
  f=f+g;
  g=f-g;
 }
}
}
```

Recursion:

```
import java.util.*;
public class fibrec {
    public static void main(String args[]) {
        int terms, i;
         Scanner in = new Scanner(System.in);
       System.out.println("Enter number of terms in Fibonacci Series");
       terms = in.nextInt();
        for(i = 0; i < terms; i++){
           System.out.print(fibonacci(i) + " ");
       }
    }
     public static int fibonacci(int num){
        if(num < 2)
            return num;
        return fibonacci(num - 1) + fibonacci(num - 2);
    }
}
```

Week 2

a) Write a Java Program that prompts the user for an integer and then prints out all the prime numbers up to that Integer.

```
import java.util.Scanner;
class PrimeNumbers
{
	public static void main(String[] args)
	{
		int n;
		int p;
		Scanner s=new Scanner(System.in);
		System.out.println("Enter a number: ");
		n=s.nextInt();
		for(int i=2;i<n;i++)
		{
			p=0;
			for(int j=2;j<i;j++)
			{
				if(i%j==0)
				p=1;
			}
			if(p==0)
				System.out.println(i);
		}
	}
}
```

b) Write a Java program that checks whether a given string is a palindrome or not.

```
import java.util.Scanner;
class ChkPalindrome
{
  public static void main(String args[])
  {
    String str, rev = "";
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter a string:");
    str = sc.nextLine();
    int length = str.length();
    for ( int i = length - 1; i >= 0; i-- )
      rev = rev + str.charAt(i);
    if (str.equals(rev))
      System.out.println(str+" is a palindrome");
    else
```

```
        System.out.println(str+" is not a palindrome");
    }
}
```

Week 3

a) Write a program to implement constructor in java.

```
class cons
{
  private String name;
  int a;
  int b;
  cons()
  {
    System.out.println("Default Constructor Called:");
  }
  cons(int a, int b)
  {
        this.a=a;
        this.b=b;
 System.out.println("Parameterizes Constructor Called with values:"+a+" and
"+b);
  }
  public static void main(String[] args) {
    cons obj = new cons();
    cons obj1 = new cons(10,14);
  }
}
```

b) Write a java program to implement method overriding.

```
class method_overriding
{
   public void eat()
   {
     System.out.println("Human is eating");
   }
}
class child extends method_overriding{
   public void eat()
   {
     System.out.println("Boy is eating");
   }
   public static void main( String args[]) {
     child obj = new child();
     obj.eat();
     method_overriding obj1 = new method_overriding();
     obj1.eat();
   }
}
```

Week 4

a) Write a Program to implement method overloading by using static method in java.

```
public class OverloadStaticMethodExample
{
public static void display()
{
System.out.println("Static method called.");
}
public static void display(int x)
{
System.out.println("An overloaded static method called.");
}
public static void main(String args[])
{
OverloadStaticMethodExample1.display();
OverloadStaticMethodExample1.display(160);
}
}
```

b) Write a Program to implement method overloading in single class in java.

Output:-

```
public class Sum {
        public int sum(int x, int y) { return (x + y); }
        public int sum(int x, int y, int z)
        {
                return (x + y + z);
        }
        public double sum(double x, double y)
        {
                return (x + y);
        }
        public static void main(String args[])
        {
                Sum s = new Sum();
                System.out.println(s.sum(10, 20));
                System.out.println(s.sum(10, 20, 30));
                System.out.println(s.sum(10.5, 20.5));
        }
}
```

Week 5

Write a java program to create an abstract class named Shape that contains two integers and an empty method named print Area (). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contains only the method print Area () that prints the area of the given shape

```
import java.util.Scanner;
abstract class Shape {
   int a = 10, b = 2;
   Shape(int a, int b){
      this.a=a;
      this.b=b;
   }
   abstract void Printarea();
}
class Rectangle extends Shape {
   Rectangle(int a, int b) {
      super(a, b);
   }
   void Printarea() {
      System.out.println("area of rectangle is " + (a * b));
   }
}
class Triangle extends Shape {
   Triangle(int a, int b) {
      super(a, b);
   }
   void Printarea(){
      System.out.println("area of triangle is " + (0.5 * a * b));
   }
}
class Circle extends Shape {
   Circle(int a, int b) {
      super(a, b);
   }
   void Printarea() {
      System.out.println("area of circle is " + (3.14 * a * a));
   }
}
class Z {
   public static void main(String[] args){
      Shape shape=null;
```

```
        String input;
        int width, height;
        while (true) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("which shape? circle/rectangle/triangle (write any
other thing for quitting): ");
            input = scanner.nextLine();
            if(!"circle".equalsIgnoreCase(input) &&
!"rectangle".equalsIgnoreCase(input) && !"triangle".equalsIgnoreCase(input)
){
                System.exit(0);
            }
            System.out.println("height: ");
            height  = scanner.nextInt();
            System.out.println("width: ");
            width = scanner.nextInt();
            if("circle".equalsIgnoreCase(input)){
                shape=new Circle(width, height);
            }
            else if("rectangle".equalsIgnoreCase(input)){
                shape=new Rectangle(width, height);
            }
            else{ // == triangle
                shape=new Triangle(width, height);
            }
            shape.Printarea();
        }
    }
}
```

Week 6

a) WAP to design a class account using the inheritance and static that show all function of bank (withdraw, deposit etc).

```
import java.util.Scanner;
class BankDetails {
   private String accno;
   private String name;
   private String acc_type;
   private long balance;
   Scanner sc = new Scanner(System.in);
   public void openAccount() {
      System.out.print("Enter Account No: ");
      accno = sc.next();
      System.out.print("Enter Account type: ");
      acc_type = sc.next();
      System.out.print("Enter Name: ");
      name = sc.next();
      System.out.print("Enter Balance: ");
      balance = sc.nextLong();
   }
   public void showAccount() {
      System.out.println("Name of account holder: " + name);
      System.out.println("Account no.: " + accno);
      System.out.println("Account type: " + acc_type);
      System.out.println("Balance: " + balance);
   }
   public void deposit() {
      long amt;
      System.out.println("Enter the amount you want to deposit: ");
      amt = sc.nextLong();
      balance = balance + amt;
   }
   public void withdrawal() {
      long amt;
      System.out.println("Enter the amount you want to withdraw: ");
      amt = sc.nextLong();
      if (balance >= amt) {
         balance = balance - amt;
         System.out.println("Balance after withdrawal: " + balance);
      } else {
         System.out.println("Your balance is less than " + amt + "\tTransaction
failed...!!" );
      }
```

```
    }
    public boolean search(String ac_no) {
        if (accno.equals(ac_no)) {
            showAccount();
            return (true);
        }
        return (false);
    }
    public static void main(String arg[]) {
        Scanner sc = new Scanner(System.in);
        //create initial accounts
        System.out.print("How many number of customers do you want to input?
");
        int n = sc.nextInt();
        BankDetails C[] = new BankDetails[n];
        for (int i = 0; i < C.length; i++) {
            C[i] = new BankDetails();
            C[i].openAccount();
        }
        int ch;
        do {
            System.out.println("\n ***Banking System Application***");
            System.out.println("1. Display all account details \n 2. Search by
Account number\n 3. Deposit the amount \n 4. Withdraw the amount \n 5.Exit
");
            System.out.println("Enter your choice: ");
            ch = sc.nextInt();
                switch (ch) {
                    case 1:
                        for (int i = 0; i < C.length; i++) {
                            C[i].showAccount();
                        }
                        break;
                    case 2:
                        System.out.print("Enter account no. you want to search: ");
                        String ac_no = sc.next();
                        boolean found = false;
                        for (int i = 0; i < C.length; i++) {
                            found = C[i].search(ac_no);
                            if (found) {
                                break;
                            }
                        }
```

```
                if (!found) {
                    System.out.println("Search failed! Account doesn't exist..!!");
                }
                break;
            case 3:
                System.out.print("Enter Account no. : ");
                ac_no = sc.next();
                found = false;
                for (int i = 0; i < C.length; i++) {
                    found = C[i].search(ac_no);
                    if (found) {
                        C[i].deposit();
                        break;
                    }
                }
                if (!found) {
                    System.out.println("Search failed! Account doesn't exist..!!");
                }
                break;
            case 4:
                System.out.print("Enter Account No : ");
                ac_no = sc.next();
                found = false;
                for (int i = 0; i < C.length; i++) {
                    found = C[i].search(ac_no);
                    if (found) {
                        C[i].withdrawal();
                        break;
                    }
                }
                if (!found) {
                    System.out.println("Search failed! Account doesn't exist..!!");
                }
                break;
            case 5:
                System.out.println("See you soon...");
                break;
            }
        }
        while (ch != 5);
    }

}
```

b) WAP to design a String class that performs String methods (Equal, Reverse the string, and change the case etc).

```
public class StringMethodsDemo {
	public static void main(String[] args) {
		String targetString = "Java is fun to learn";
		String s1= "JAVA";
		String s2= "Java";
		String s3 = "  Hello Java  ";
		System.out.println("Char at index 2(third position): " +
targetString.charAt(2));
		System.out.println("After Concat: "+ targetString.concat("-Enjoy-
"));
		System.out.println("Checking equals ignoring case: "
+s2.equalsIgnoreCase(s1));
		System.out.println("Checking equals with case: " +s2.equals(s1));
		System.out.println("Checking Length: "+ targetString.length());
		System.out.println("Replace function: "+
targetString.replace("fun", "easy"));
		System.out.println("SubString of targetString: "+
targetString.substring(8));
		System.out.println("SubString of targetString: "+
targetString.substring(8, 12));
		System.out.println("Converting to lower case: "+
targetString.toLowerCase());
		System.out.println("Converting to upper case: "+
targetString.toUpperCase());
		System.out.println("Triming string: " + s3.trim());
		System.out.println("searching s1 in targetString: " +
targetString.contains(s1));
		System.out.println("searching s2 in targetString: " +
targetString.contains(s2));

		char [] charArray = s2.toCharArray();
		System.out.println("Size of char array: " + charArray.length);
		System.out.println("Printing last element of array: " +
charArray[3]);

	}

}
```

Week 7

a) WAP to create a package that access the member of external class as well as same package.

Package 1:

Pack1.java

```
package pack1;
interface pack1Interface {
	String name = "This is the Interface of pack1";
	void pack1Interface();
}
public class pack1 {
		String name;
	public String getName() { return name; }
	public void setName(String name) { this.name = name; }
}
```

Package 2:

Pack3.java

```
package pack2;
interface pack3Interface {
	String name = "pack";
	public void interfacepack();
}
abstract class packabstract {
	String name = "packAbstract";
	abstract public void print();
}
public class pack3 {
		int first;
	int second;
	pack3(int a, int b)
	{
		this.first = a;
		this.second = b;
	}
	public int add() { return this.first + this.second; }
}
```

Accessing the members of package 1 class in package 2 class:

Pack2.java

```
package pack2;
import pack1.*;
public class pack2 implements pack3Interface {

	@Override public void interfacepack()
	{
		System.out.println(
			"This is the interface of the pack3class");
	}
	public static void main(String args[])
	{
		pack1 ob = new pack1();
		ob.setName("packsetter");
		System.out.println(ob.getName());
		pack2 ob2 = new pack2();
		ob2.interfacepack();
	}
}
```

b) WAP that import the user define package and access the member variable of classes that contained by package.

Note:- Create Pack.java in udp directory

```
package udp;
public class pack
{
public int a=20;
public pack()
{
 System.out.println("Constructor of mypack");
}
public void mypackm1()
{
System.out.println("I am ny packm1");
}
}
```

Packex.java

```
import udp.pack;
class packex
{
```

```
  public static void main(String[] args) {
    pack obj = new pack();
    System.out.println(obj.a);
    obj.mypackm1();
  }
}
```

Week 8

a) Write a Java program that describes exception handling mechanism.

```
class TestException {
  public static void main(String args[]){
  try{
   int data=25/5;
   System.out.println(data);
  }
  catch(NullPointerException e){
System.out.println(e);
}
 finally {
System.out.println("finally block is always executed");
}
System.out.println("rest of phe code...");
  }
}
```

b) Write a java program Illustrate multiple catch clauses.

```
    public class MultipleCatchBlock1 {
            public static void main(String[] args) {
                    try{
                            int a[]=new int[5];
                            a[5]=30/0;
                           }
                           catch(ArithmeticException e)
                        {
                                 System.out.println("Arithmetic Exception occurs");
                                }
                           catch(ArrayIndexOutOfBoundsException e)
                        {
                    System.out.println("ArrayIndexOutOfBounds Exception occurs");
                                }
                           catch(Exception e)
                        {
                                 System.out.println("Parent Exception occurs");
                                }
                           System.out.println("rest of the code");
            }
    }
```

Weeks 9

Write a java program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

```
import java.util.Random;
class Square extends Thread
{
 int x;
 Square(int n)
 {
 x = n;
 }
 public void run()
 {
 int sqr = x * x;
 System.out.println("Square of " + x + " = " + sqr );
 }
}
class Cube extends Thread
{
 int x;
 Cube(int n)
 {
 x = n;
 }
 public void run()
 {
 int cub = x * x * x;
 System.out.println("Cube of " + x + " = " + cub );
 }
}
class Number extends Thread
{
 public void run()
 {
 Random random = new Random();
 for(int i =0; i<10; i++)
 {
 int randomInteger = random.nextInt(100);
 System.out.println("Random Integer generated : " + randomInteger);
 Square s = new Square(randomInteger);
```

```
 s.start();
 Cube c = new Cube(randomInteger);
 c.start();
 try {
 Thread.sleep(1000);

 } catch (InterruptedException ex) {
 System.out.println(ex);
}
 }
 }
}
public class week9 {
 public static void main(String args[])
 {
 Number n = new Number();
 n.start();
 }
}
```

Week 10

a) Write a JAVA program Producer Consumer Problem

b) Write a case study on thread Synchronization after solving the above producer consumer problem

```
import java.util.LinkedList;
public class Threadexample {
   public static void main(String[] args)
      throws InterruptedException
   {
      final PC pc = new PC();
      Thread t1 = new Thread(new Runnable() {
         @Override
         public void run()
         {
            try {
               pc.produce();
            }
            catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
      });
      Thread t2 = new Thread(new Runnable() {
         @Override
         public void run()
         {
            try {
               pc.consume();
            }
            catch (InterruptedException e) {
               e.printStackTrace();
            }
         }
      });
      t1.start();
      t2.start();
      t1.join();
      t2.join();
   }
   public static class PC {
       LinkedList<Integer> list = new LinkedList<>();
      int capacity = 2;
      public void produce() throws InterruptedException
```

```
        {
            int value = 0;
            while (true) {
                synchronized (this)
                {
                    while (list.size() == capacity)
                        wait();
                    System.out.println("Producer produced-"+ value);
                    list.add(value++);
                    notify();
                    Thread.sleep(1000);
                }
            }
        }
        public void consume() throws InterruptedException
        {
            while (true) {
                synchronized (this)
                {
                    while (list.size() == 0)
                        wait();
                    int val = list.removeFirst();
                     System.out.println("Consumer consumed-"+ val);
                     notify();
                    Thread.sleep(1000);
                }
            }
        }
    }
}
```

Week 11:

a) Write a java program that reads a file name from the user, and then displays information about whether the file exists, whether the file is readable, whether the file is writable, the type of file and the length of the file in bytes.

```
import java.io.*;
class filedemo
{
public static void p(String str)
{
   System.out.println(str);
 }
 public static void analyze(String s)
File f=new File(s);
 if(f.exists())
 {
 p(f.getName()+" is a file");
 p(f.canRead()?" is readable":" is not readable");
 p(f.canWrite()?" is writable":" is not writable");
 p("Filesize:"+f.length()+" bytes");
 p("File last mdified:"+f.lastModified());
 }
 if(f.isDirectory())
 {
 p(f.getName()+" is directory");
 p("List of files");
 String dir[]=f.list();
 for(int i=0;i<dir.length;i++)
 p(dir[i]);
 }
 }
}
public class FileDetails
{
 public static void main(String rr[])throws IOException
 {
 filedemo fd=new filedemo();
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
 System.out.println("Enter the file name:");
 String s=br.readLine();
 fd.analyze(s);
```

```
 }
}
```

b) Write a java program that displays the number of characters, lines and words in a text file.

```
import java.io.*;
public class Test {
	public static void main(String[] args)
		throws IOException
	{
		File file = new File("C:\\Users\\hp\\Desktop\\TextReader.txt");
		FileInputStream fileInputStream = new FileInputStream(file);
		InputStreamReader inputStreamReader = new
InputStreamReader(fileInputStream);
		BufferedReader bufferedReader = new
BufferedReader(inputStreamReader);
		String line;
		int wordCount = 0;
		int characterCount = 0;
		int paraCount = 0;
		int whiteSpaceCount = 0;
		int sentenceCount = 0;

		while ((line = bufferedReader.readLine()) != null) {
			if (line.equals("")) {
				paraCount += 1;
			}
			else {
				characterCount += line.length();
				String words[] = line.split("\\s+");
				wordCount += words.length;
				whiteSpaceCount += wordCount - 1;
				String sentence[] = line.split("[!?.:]+");
				sentenceCount += sentence.length;
			}
		}
		if (sentenceCount >= 1) {
			paraCount++;
		}
		System.out.println("Total word count = "+ wordCount);
		System.out.println("Total number of sentences = "+
sentenceCount);
```

```
                System.out.println("Total number of characters = "+
characterCount);
                System.out.println("Number of paragraphs = "+ paraCount);
                System.out.println("Total number of whitespaces = "+
whiteSpaceCount);
        }
}
```

Week 12:

a) Write a Java Program to create ArrayList and retrieve the ArrayListItems using iterators.

```
import java.util.ArrayList;
class Main {
  public static void main(String[] args) {
    ArrayList<String> languages = new ArrayList<>();
    languages.add("Java");
    languages.add("JavaScript");
    languages.add("Python");
    System.out.println("ArrayList: " + languages);
    System.out.println("Iterating over ArrayList using for loop: ");
    for(int i = 0; i < languages.size(); i++) {
      System.out.print(languages.get(i));
      System.out.print(", ");
    }
  }
}
```

b) Write a Java Program to perform stack operation using Collections

```
import java.io.*;
import java.util.*;
class Test
{
        static void stack_push(Stack<Integer> stack)
        {
                for(int i = 0; i < 5; i++)
                {
                        stack.push(i);
                }
```

```
}
static void stack_pop(Stack<Integer> stack)
{
        System.out.println("Pop Operation:");

        for(int i = 0; i < 5; i++)
        {
                Integer y = (Integer) stack.pop();
                System.out.println(y);
        }
}
static void stack_peek(Stack<Integer> stack)
{
        Integer element = (Integer) stack.peek();
        System.out.println("Element on stack top: " + element);
}
static void stack_search(Stack<Integer> stack, int element)
{
        Integer pos = (Integer) stack.search(element);
        if(pos == -1)
                System.out.println("Element not found");
        else
                System.out.println("Element is found at position: " + pos);
}
public static void main (String[] args)
{
        Stack<Integer> stack = new Stack<Integer>();
        stack_push(stack);
        stack_pop(stack);
```

```
            stack_push(stack);
            stack_peek(stack);
            stack_search(stack, 2);
            stack_search(stack, 6);
        }
}
```