ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT

(An Autonomous Institution)

TEKKALI

Department of MCA

LAB MANUAL

MCA-AR 24 Regulations –I Year II Semester

JAVA Programming Lab(24MCA2012)

Prepared by

Dr.Naresh Tangudu, Associate Professor

Index

Aim: Write a Java Program that uses both recursive and non-recursive functions to print the nth value of the Fibonacci sequence.

Code:

```
import java.util.Scanner;

class Main
{
   public static void main(String args[])
   {
      System.out.println("Enter the number n to print the faboniccs series ");
      Scanner ob=new Scanner(System.in);
      short a=ob.nextShort();
      Series ob1=new Series();
      long b=ob1.input(a);
      System.out.println("The "+a+"th number of the faboniccs series is "+b);
   }
}

class Series
{

   int a=1;
   int b=1;
   int c=0;
   int count;
   int input(int a)
   {
      count=a;
      count=fabo(count);
      return count;
   }

   int fabo(int count)
   {
      if(count!=2)
      {
         c=a+b;
         a=b;
         b=c;
         fabo(--count);
      }
      return c;
   }
}
```

Input-Output

1)

```
Enter the number n to print the faboniccs series
5
```

```
The 5th number of the faboniccs series is 5
2)
Enter the number n to print the faboniccs series
9
The 9th number of the faboniccs series is 34
3)
Enter the number n to print the faboniccs series
12
The 12th number of the faboniccs series is 144
```

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

Code:

```java
import java.util.Scanner;

class Main
{
	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);
		}
	}
}
```

Input-Output

1)

```
Enter a number:
10
2
3
5
7
```

2)

```
Enter a number:
20
2
3
5
7
11
13
17
19
```

Aim: Write a Java Program that checks whether a given string is a palindrome or not. Ex. MALAYALAM is a palindrome.

Code:

```java
import java.util.Scanner;
class Main {
  public static void main(String[] args) {

    String str ="", reverseStr = "";
    System.out.println("enter the string");
    Scanner stringScanner = new Scanner(System.in);
    str = stringScanner.next();

    int strLength = str.length();

    for (int i = (strLength - 1); i >=0; --i) {
      reverseStr = reverseStr + str.charAt(i);
    }

    if (str.toLowerCase().equals(reverseStr.toLowerCase())) {
      System.out.println(str + " is a Palindrome String.");
    }
    else {
      System.out.println(str + " is not a Palindrome String.");
    }
  }
}
```

Input-Output

1)

```
enter the string
madam
madam is a Palindrome String.
```

2)

```
enter the string
naresh
naresh is not a Palindrome String.
```

3)

```
enter the string
Malayalam
Malayalam is a Palindrome String.
```

Aim: Write a Java Program to implement abstraction and encapsulation.

Code:

```
// Abstract class (Abstraction)
abstract class BankAccount {
   private double balance; // Encapsulation

   // Constructor
   public BankAccount(double balance) {
      this.balance = balance;
   }

   // Getter method (Encapsulation)
   public double getBalance() {
      return balance;
   }

   // Method to deposit money
   public void deposit(double amount) {
      balance += amount;
   }

   // Abstract method (Abstraction)
   abstract void withdraw(double amount);
}

// Concrete class implementing abstraction
class SavingsAccount extends BankAccount {
   public SavingsAccount(double balance) {
      super(balance);
   }

   // Implementing abstract method
   public void withdraw(double amount) {
      if (amount <= getBalance()) {
         deposit(-amount);
         System.out.println("Withdrawn: $" + amount);
      } else {
         System.out.println("Insufficient balance!");
      }
   }
}

// Main class
public class Main {
   public static void main(String[] args) {
      SavingsAccount account = new SavingsAccount(1000);
      System.out.println("Balance: $" + account.getBalance());
      account.deposit(500);
```

```
        account.withdraw(300);
        System.out.println("Final Balance: $" + account.getBalance());
    }
}
```

Input-Output

```
Balance: $1000.0
Withdrawn: $300.0
Final Balance: $1200.0
```

Aim: Write a Java Program to implement multiple inheritance.

Code

```
interface Backend {

  // abstract class
  public void connectServer();
}

class Frontend {

  public void responsive(String str) {
    System.out.println(str + " can also be used as frontend.");
  }
}

// Language extends Frontend class
// Language implements Backend interface
class Main extends Frontend implements Backend {

  String language = "Java";

  // implement method of interface
  public void connectServer() {
    System.out.println(language + " can be used as backend language.");
  }

  public static void main(String[] args) {

    // create object of Language class
    Main java = new Main();

    java.connectServer();

    // call the inherited method of Frontend class
    java.responsive(java.language);
  }
```

Input-output

```
Java can be used as backend language.
Java can also be used as frontend.
```

Aim: Write a java program to illustrate method overloading and method overriding, covariant sub typing.

Code

```
// Method Overloading Example
class MathOperations {
   // Overloaded methods
   public int add(int a, int b) {
      return a + b;
   }

   public double add(double a, double b) {
      return a + b;
   }
}

// Parent class
class Animal {
   // Overridden method (to be overridden in Dog)
   Animal makeSound() {
      System.out.println("Animal makes a sound");
      return new Animal();
   }
}

// Subclass demonstrating method overriding & covariant return type
class Dog extends Animal {
   // Overriding method with covariant return type (returning Dog instead of Animal)
   @Override
   Dog makeSound() {
      System.out.println("Dog barks");
      return new Dog();
   }
}

// Main class to test the program
public class Main {
   public static void main(String[] args) {
      // Method Overloading
      MathOperations math = new MathOperations();
      System.out.println("Addition (int): " + math.add(5, 10));
      System.out.println("Addition (double): " + math.add(5.5, 2.2));

      // Method Overriding & Covariant Return Type
      Animal myAnimal = new Animal();
      myAnimal.makeSound();

      Animal myDog = new Dog(); // Upcasting
      myDog.makeSound();
```

```
    }
}
```

Input-Output

```
Addition (int): 15
Addition (double): 7.7
Animal makes a sound
Dog barks
```

Aim: Write a java program that illustrates how java achieved Run Time Polymorphism

Code:

```
class Bike{
  void run(){System.out.println("running");}
}
class Splendor extends Bike{
  void run(){System.out.println("running safely with 60km");}
}
class Main{

  public static void main(String args[]){
    Bike b = new Splendor();//upcasting
    b.run();
  }
}
```

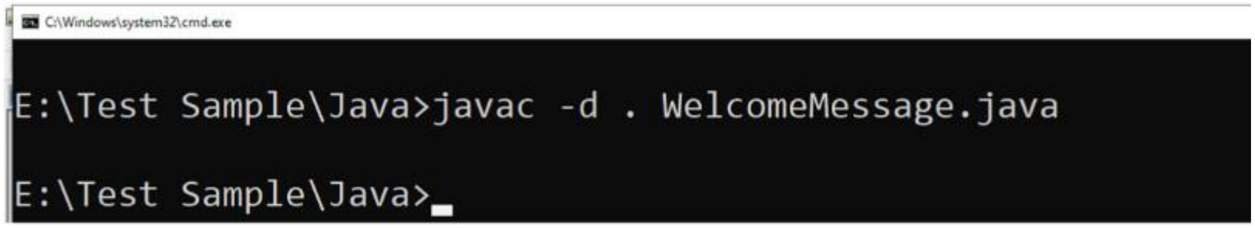
Input-output

```
running safely with 60km
```

Aim: Write a Java Program to create and demonstrate packages.

Code

```
package university;
public class WelcomeMessage
{
//has one method ShowMessage()
public void ShowMessage()
{
System.out.println("Welcome to our University");
}
}
```



```
E:\Test Sample\Java>javac -d . WelcomeMessage.java

E:\Test Sample\Java>_
```

```
Import university.*;
Class Main
{
Public static void main(String a[])
{
WelcomeMessage w1=new WelcomeMessage();
w1. ShowMessage();


}
}
```

Input-Output

```
Welcome to our University
```

Aim: Write a Java program to implement the concept of importing classes from user defined package and creating packages

Code:

```
// Define package
package mypackage;

// User-defined class in a package
public class MyClass {
    public void displayMessage() {
        System.out.println("Hello from MyClass in mypackage!");
    }
}

// Importing user-defined package
import mypackage.MyClass;

public class Main {
    public static void main(String[] args) {
        // Creating an object of MyClass from mypackage
        MyClass obj = new MyClass();
        obj.displayMessage(); // Calling method from imported class
    }
}
```

Open the terminal or command prompt.

Navigate to the directory containing your `mypackage` folder and `Main.java`.

Compile

```
javac -d . mypackage/MyClass.java
javac Main.java
java Main
```

Input-Output

```
Hello from MyClass in mypackage!
```

Aim: Write a java program to implement the concept of Exception Handling by using predefined and user defined exceptions.

Code:

```
// User-defined exception class
class InvalidAgeException extends Exception {
   public InvalidAgeException(String message) {
      super(message);
   }
}

public class Main {
   public static void main(String[] args) {
      // Handling predefined exception (ArithmeticException)
      try {
         int result = 10 / 0; // Division by zero
      } catch (ArithmeticException e) {
         System.out.println("Error: Division by zero is not allowed!");
      }

      // Handling user-defined exception
      try {
         checkAge(16); // This will throw InvalidAgeException
      } catch (InvalidAgeException e) {
         System.out.println("Custom Exception: " + e.getMessage());
      }
   }

   // Method that throws a user-defined exception
   public static void checkAge(int age) throws InvalidAgeException {
      if (age < 18) {
         throw new InvalidAgeException("Age must be 18 or above.");
      }
      System.out.println("You are eligible.");
   }
}
```

Input-Output

```
ERROR!
Error: Division by zero is not allowed!
Custom Exception: Age must be 18 or above.
```

Aim: Write a Java Program demonstrating the life cycle of a thread

Code:

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is RUNNING...");

        try {
            // Thread goes to TIMED_WAITING state
            Thread.sleep(2000);
            System.out.println("Thread is in TIMED_WAITING state (sleep).");
        } catch (InterruptedException e) {
            System.out.println("Thread was interrupted.");
        }

        System.out.println("Thread is TERMINATED.");
    }
}

public class Main {
    public static void main(String[] args) {
        // NEW state: Thread is created
        MyThread t = new MyThread();
        System.out.println("Thread is in NEW state.");

        // RUNNABLE state: Thread starts
        t.start();
        System.out.println("Thread is in RUNNABLE state (started).");

        // Let main thread wait for MyThread to finish
        try {
            t.join();  // Ensures main waits until 't' is terminated
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        System.out.println("Main thread finished execution.");
    }
}
```

Input-output

```
Thread is in NEW state.
Thread is in RUNNABLE state (started).
Thread is RUNNING...
Thread is in TIMED_WAITING state (sleep).
Thread is TERMINATED.
Main thread finished execution.
```

Aim: Write a java program to implement thread priorities

Code:

```
class MyThread extends Thread {
    public MyThread(String name) {
        super(name); // Set thread name
    }

    public void run() {
        System.out.println(getName() + " (Priority: " + getPriority() + ") is running...");
    }
}

public class Main {
    public static void main(String[] args) {
        // Creating three threads
        MyThread t1 = new MyThread("Low Priority Thread");
        MyThread t2 = new MyThread("Normal Priority Thread");
        MyThread t3 = new MyThread("High Priority Thread");

        // Setting priorities
        t1.setPriority(Thread.MIN_PRIORITY); // Priority 1
        t2.setPriority(Thread.NORM_PRIORITY); // Priority 5 (Default)
        t3.setPriority(Thread.MAX_PRIORITY); // Priority 10

        // Starting threads
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Input-output

```
Low Priority Thread (Priority: 1) is running...
Normal Priority Thread (Priority: 5) is running...
High Priority Thread (Priority: 10) is running...
```