USN :2GI19CS175 Student Name : Venkatesh G Dhongadi

Title of the Experiment (Interfaces)

Experiment No. ____07_____ **Date :**__24/12/20_____

Problem Statement:

Write a Java application to implement the following UML diagram.

- PrimeTester class implements isPrime() method by iterating from 2 to n-1 for a given number n
- ImprPrimeTester class implements isPrime() method by iterating from 2 to n/2
- FasterPrimeTester class implements isPrime() method by iterating from 2 to
- FastestPrimeTester class implements isPrime() method using Fermat's Little theorem.
    - o Fermat's Little Theorem:
    - o If n is a prime number, then for every a, 1 < a < n-1, an-1 % n = 1

Objectives of the Experiment:

1. Learn declaration and initialization of variables and Interfaces in Java.
2. Understand the use of Interfaces in a real-life application.
3. Learn the usage of Looping constructs and control statements.
4. Learn to Display the result in a readable/proper format.

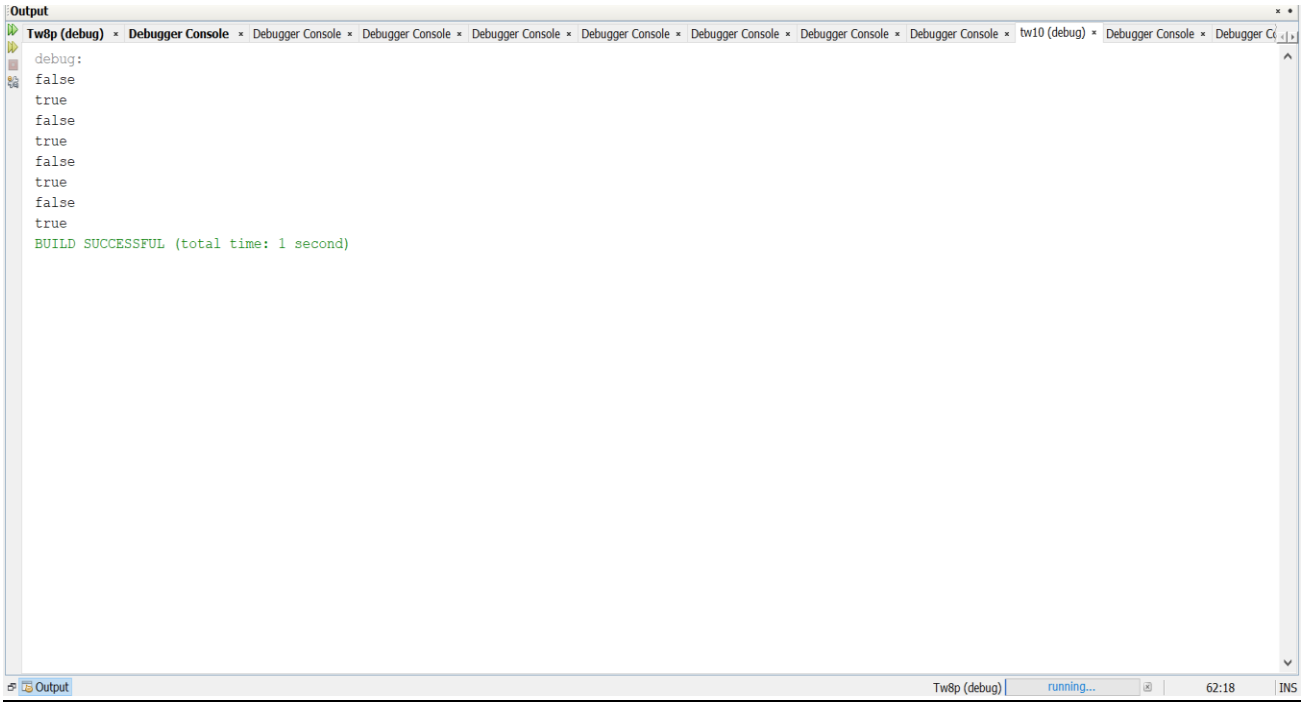
Problem Source Code:

```
package termwork_7;

publicinterfaceIPrime{
	booleanisPrime(int n);
}

package termwork_7;

classPrimeTesterimplementsIPrime{
	publicbooleanisPrime(int n) {
		boolean flag = true;
		for(inti=2; i<n; i++) {
			if(n % i == 0) {
				flag = false;
				break;
			}
		}
		return flag;
	}
}
```

```
class ImprPrimeTester implements IPrime{
	public boolean isPrime(int n) {
		boolean flag = true;
		for(int i=2; i<n/2; i++) {
			if(n % i == 0) {
				flag = false;
				break;
			}
		}
		return flag;
	}
}


class FasterPrimeTester implements IPrime{
	public boolean isPrime(int n){
		boolean flag = true;
		for(int i=2; i<Math.sqrt(n); i++) {
			if(n % i == 0) {
				flag = false;
				break;
			}
		}
		return flag;
	}

}

class FastestPrimeTester implements IPrime{
	public boolean isPrime(int n) {
		int a = 2;
		if(Math.pow(a, n-1) % n == 1) {
			returntrue;
		}
		else {
			returnfalse;
		}
	}
}

public class Prime {

	public static void main(String[] args) {
		PrimeTester p1 = new PrimeTester();
		ImprPrimeTester p2 = new ImprPrimeTester();
		FasterPrimeTester p3 = new FasterPrimeTester();
		FastestPrimeTester p4 = new FastestPrimeTester();
		System.out.println(p1.isPrime(12));
		System.out.println(p1.isPrime(13));
		System.out.println(p2.isPrime(12));
		System.out.println(p2.isPrime(13));
```

```
                System.out.println(p3.isPrime(12));
                System.out.println(p3.isPrime(13));
                System.out.println(p4.isPrime(12));
                System.out.println(p4.isPrime(13));
        }

}
```

Output:



Outcomes of the Experiment:

1. Able to Demonstrate the use of Interfaces in solving real-life problems.
2. Identify appropriate variables and their types
3. Identify appropriate looping constructs (for)
4. Check if one loop will suffice or use nesting
5. Identify the control statements needed to meet the problem requirements.

Conclusions:

From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two sets of input and result obtained were verified to be correct and recorded.

Practice Problem Statement:

Write a JAVA program which has:
i. An Interface class for Stack Operations (viz., push(), pop(), peek(),display())
ii. A Class that implements the Stack Interface and creates a fixed length Stack.
iii. A Class that implements the Stack Interface and creates a Dynamic Length Stack.
iv. A Class that uses both the above Stacks through Interface reference and does the Stack operations that demonstrates the runtime binding.

Problem Source Code:

```
package termwork_7_pp1;

publicinterfaceStack{
	voidpush(int e);
	voidpop();
	voidpeek();
	voiddisplay();
}

package termwork_7_pp1;
importjava.util.ArrayList;

classFStackimplementsStack{
	int [] element;
	int top;

	FStack(int size){
		element = newint [size];
		top = -1;
	}

	publicvoidpush(int e) {
		if(top == element.length-1) {
			System.out.println("Stack Overflow");
		}
		else {
			element[++top] = e;
		}
	}

	publicvoidpop() {
		if(top == -1) {
			System.out.println("Stack Underflow");
		}
		else {
			System.out.println(element[top--] + " is popped");
		}
```

```
	}

	publicvoidpeek() {
		if(top == -1) {
			System.out.println("No elements in the Stack");
		}
		else {
			System.out.println(element[top] + " is on top of the Stack");
		}
	}

	publicvoiddisplay() {
		System.out.println("The status of the Stack: ");
		for(inti=top; i>= 0; i--) {
			System.out.println(element[i]);
		}
	}
}

classDStackimplementsStack{
	ArrayList<Integer>element;
	int top;

	DStack(){
		top = 0;
		element = newArrayList();
	}

	publicvoidpush(int e) {
		element.add(e);
		top++;
	}

	publicvoidpop() {
		if(element.size() == 0) {
			System.out.println("Stack Underflow");
		}
		else {
			System.out.println(element.remove(--top) + " is popped");
		}
	}

	publicvoidpeek() {
		if(element.size() == 0) {
			System.out.println("No elements in the Stack");
		}
		else {
			System.out.println(element.get(top-1) + " is on top of the Stack");
		}
	}
```

```
        publicvoiddisplay() {
                System.out.println("The status of the Stack: ");
                for(inti=element.size()-1; i>= 0; i--) {
                        System.out.println(element.get(i));
                }
        }
}

publicclass tw7a {
        publicstaticvoidmain(String[] args) throws Exception{
                DStack ds = newDStack();
                ds.push(10);
                ds.push(20);
                ds.push(30);
                ds.push(40);
                ds.push(50);
                ds.display();
                ds.peek();
                ds.pop();
                ds.pop();
                ds.pop();
                ds.pop();
                ds.pop();
        }
}
```

Output: