

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 1

Use Java compiler and Eclipse platform to write and execute java program

Practical 1

Program Code:

```
class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, Java World!");  
    }  
}
```

Input and Output:

```
> javac HelloWorld.java
```

```
> java HelloWorld
```

```
Hello, Java World!
```

Page 1

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 2

Creating simple Java program using command line arguments

Practical 2

Program Code:

```
class CommandLineDemo {  
    public static void main(String[] args) {  
        System.out.println("You entered: " + args[0]);  
    }  
}
```

Input and Output:

```
> javac CommandLineDemo.java
```

```
> java CommandLineDemo Hello
```

You entered: Hello

Page 2

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 3

Understand OOP concepts and basics of Java programming

Practical 3

Program Code:

```
class Student {  
    String name;  
    int roll;  
    void insert(String n, int r) {  
        name = n;  
        roll = r;  
    }  
    void display() {  
        System.out.println("Name: " + name + ", Roll: " + roll);  
    }  
    public static void main(String[] args) {  
        Student s1 = new Student();  
        s1.insert("Ayush", 101);  
        s1.display();  
    }  
}
```

Input and Output:

```
> javac Student.java
```

```
> java Student
```

Name: Ayush, Roll: 101

Page 3

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 4

Create Java programs using inheritance and polymorphism

Practical 4

Program Code:

```
class Animal {  
void sound() {  
System.out.println("Animal makes sound");  
}  
}  
class Dog extends Animal {  
void sound() {  
System.out.println("Dog barks");  
}  
public static void main(String[] args) {  
Animal a = new Dog(); // Polymorphism  
a.sound();  
}  
}
```

Input and Output:

> javac Animal.java

> java Dog

Dog barks

Page 4

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 5

Implement error-handling techniques using exception handling and multithreading

Practical 5

Program Code:

```
class MyThread extends Thread {  
    public void run() {  
        System.out.println("Thread is running...");  
    }  
    public static void main(String[] args) {  
        try {  
            int data = 50 / 0;  
        } catch (ArithmeticException e) {  
            System.out.println("Exception caught: " + e);  
        }  
        MyThread t1 = new MyThread();  
        t1.start();  
    }  
}
```

Input and Output:

> javac MyThread.java

> java MyThread

Exception caught: java.lang.ArithmeticException: / by zero

Thread is running...

Page 5

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 6

Create Java program with the use of java packages

Practical 6

Program Code:

```
package mypack;
public class MyPackageClass {
    public void display() {
        System.out.println("This is my package!");
    }
}

// Save in MyPackageClass.java and compile with: javac -d . MyPackageClass.java
// Then create another file to use it:
import mypack.MyPackageClass;
class TestPackage {
    public static void main(String[] args) {
        MyPackageClass obj = new MyPackageClass();
        obj.display();
    }
}
```

Input and Output:

```
> javac -d . MyPackageClass.java
```

```
> javac TestPackage.java
```

```
> java TestPackage
```

This is my package!

OBJECT ORIENTED PROGRAMMING WITH JAVA (BCS-452)

Practical 7

Construct java program using java I/O package

Practical 7

Program Code:

```
import java.io.*;
class FileWriteRead {
public static void main(String[] args) {
try {
FileWriter fw = new FileWriter("test.txt");
fw.write("Hello File");
fw.close();
FileReader fr = new FileReader("test.txt");
int i;
while ((i = fr.read()) != -1)
System.out.print((char) i);
fr.close();
} catch (IOException e) {
System.out.println(e);
}
}
}
```

Input and Output:

```
> javac FileWriteRead.java
```

```
> java FileWriteRead
```

Hello File

Page 7