SDM COLLEGE OF ENGINEERING AND TECHNOLOGY

Dhavalagiri,Dharwad-580002, Karnataka State, India.

Email: cse.sdmcet@gmail.com

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Report

On

Minor work Task-3

COURSE CODE : 22UHUC500

COURSE TITLE :Software Engineering and Project Management

SEMISTER : 5th DIVISION : CSE 'B'

COURSE TEACHER : Dr. U.P Kulkarni

[Academic Year- 2024-25]

Date of Submission: 09-10-2024

Submitted

By

Mr. Sumpreetsing S Rajaput USN : 2SD22CS112

Contents

Problem Statement: “List the features of programming language and write programs to show how they help to write robust code.”

Introduction:

A programming language is a formal set of instructions used to communicate with computers to perform specific tasks. It allows developers to write code that the machine can interpret and execute.

Here are some of the Key features of Programming languages

1. Strong Typing
2. Error Handling
3. Modularity
4. Code Reusability
5. Standard Libraries

Illustration of Features :

1. Strong Typing :

Strong typing in programming languages refers to the enforcement of strict rules on how different types of data can interact with one another. In strongly typed languages, each variable, expression, and value has a specific data type (such as integer, float, string, etc.), and operations involving mismatched data types are either disallowed or require explicit conversion. This feature helps catch errors at compile-time or runtime, reducing bugs caused by unintended type conversions or misuse of data types

Here is a java code that demonstrates **Strong Typing**, where operations between mismatched data types require explicit handling, preventing unintended errors.

```java
public class Main {
    public static void main(String[] args) {
        int number = 10;
        String text = "Hello";

        // This line will cause a compile-time error because of type mismatch:
        // String result = number + text; // Error: incompatible types: int
cannot be converted to String

        // Correct approach: explicit conversion of int to String
        String result = text + Integer.toString(number); // Explicit
conversion of int to String
        System.out.println(result);  // Output: Hello10

    }
}
```

Fig 1:Strong Typing

2. Error Handling :

Error handling in programming refers to the methods and practices used to manage and respond to errors that occur during the execution of a program. Errors can arise from various sources, such as invalid user input, accessing unavailable resources, or attempting operations that the system cannot handle (like division by zero or accessing out-of-bounds array elements).

Here is a java code that demonstrates **Error Handling** feature

```
public class Main {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3};

        try {
            // This line will throw an ArrayIndexOutOfBoundsException
            System.out.println(numbers[5]);  // Trying to access index 5,
which is out of bounds
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Tried to access an invalid index.");
        } finally {
            System.out.println("This block always executes, even if there's an
exception.");
        }

        System.out.println("Program continues normally after handling the
error.");
    }
}
```

Fig 2: Error Handling

3. Modularity :

Modularity in programming refers to the design principle of breaking down a large program into smaller, independent, and manageable units or modules. Each module performs a specific part of the overall functionality and can be developed, tested, and maintained separately. Modularity promotes code reuse, ease of debugging, maintainability, and scalability.

Here ia the java code that demonstrates how it supports modular programming through classes and methods.

```java
public class Main {
    public static double calculateArea(double radius) {
        return Math.PI * radius * radius;
    }

    public static double calculateCircumference(double radius) {
        return 2 * Math.PI * radius;
    }

    public static void main(String[] args) {
        double radius = 5.0;
        System.out.println("Area: " + calculateArea(radius)); // Output: Area: 78.53981633974483
        System.out.println("Circumference: " + calculateCircumference(radius)); // Output: Circumference: 31.41592653589793
    }
}
```

Fig 3:Modularity

4. Code Reusability :

Code reusability is a fundamental principle in software development that emphasizes the practice of using existing code in multiple applications or parts of a program without modification. It allows developers to avoid redundancy, reduce errors, and save time by leveraging previously written and tested code.

Here is the java code that tells how methods in it are reused across the different parts of code

```java
public class Main {
    public static void greetUser(String name) {
        System.out.println("Hello, " + name + "!");
    }
    public static void main(String[] args) {
        greetUser("Alice"); // Output: Hello, Alice!
        greetUser("Bob");   // Output: Hello, Bob! }
}
```

Fig 4:Code Reusability

5. Standard Libraries :

Standard libraries are a collection of pre-written code and functions provided by a programming language to perform common tasks. These libraries offer a wide range of functionalities that facilitate development, allowing programmers to write code more efficiently without needing to implement everything from scratch. Standard libraries cover various areas, including data structures, algorithms, file handling, networking, and graphical user interfaces.

Here is the code that demonstrates how java provides a rich set of standard libraries that reduse the need for custome implementation

```java
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Cherry");
        System.out.println(fruits); // Output: [Apple, Banana, Cherry]
    }
}
```

Fig 5: Standard Libraries

Conclusion :

In conclusion, the features of programming languages, such as **Strong Typing**, **Error Handling**, **Modularity**, **Code Reusability**, and **Standard Libraries**, collectively contribute to the development of robust, efficient, and maintainable software.