**Tutorial -2**

Question:

1. Define the pure function/method. Compare it to impure function/method.

|  |  |  |
| --- | --- | --- |
|  | Pure Function | Impure Function |
| Definition | A function that uses only its arguments as inputs, without access to global variables or external factors. | A function that may use or alter global variables, rely on external factors or cause side effects. |
| Side Effects | No Side Effects. The function does not change any state or data, it only computes and returns a value. | Have Side Effects. The function can change state or data which affects the program outside of function. |
| Global Variables | Does not use global variables. All data is needed to pass through the function’s arguments. | Use or alter global variables. |
| Referential Transparency | Exhibits referential transparency. The function will always create the same output based on the same input. | Not necessary exhibit referential transparency. The function may create different output even if it is the same input due to external factors or global variables. |

1. Appraise the concept of “side effects”. Explain TWO (2) side effects in any programming language with code examples.

|  |  |  |
| --- | --- | --- |
| Side Effect | Description | Code Example |
| Modifying a global variable | A function changes the value of a global variable. |  |
| Writing to a file | A function writes data to a file or another external resource. |  |

1. Given the Java code as follows. Explain the effect of code design.

|  |
| --- |
| public class Demo {    public static void main(String[] args) {    //demo  Circle c1 = new Circle();  c1.setRadius( 10 );  System.out.println( "=== Area, Circumference, Diameter ===" );  c1.area();  System.out.println( "Area: " + c1.area);  c1.circumference();  System.out.println( "Circumference: " + c1.circumference);  c1.diameter();  System.out.println( "Diameter: " + c1.diameter);    }  }  class Circle{    int radius;  double area;  double circumference;  double diameter;    //getter  public void setRadius(int radius) {  this.radius = radius;  }    public void area() {  if( radius > 0 ) {  this.area = Math.PI \* Math.pow(radius, 2);  }  }    public void circumference() {  if( radius > 0 ) {  this.circumference = 2 \* Math.PI \* radius;  }  }    public void diameter() {  if( radius > 0 ) {  this.diameter = 2 \* radius;  }  }    } |

|  |  |  |
| --- | --- | --- |
| **Code Design Principle** | **Effect** | **Example** |
| Encapsulation | The Circle class encapsulates the properties of a circle (radius, area, circumference, diameter) and the methods that operate on these properties (setRadius, area, circumference, diameter). This design makes the code easier to understand and maintain, as each class is responsible for its own state and behavior. | Circle class in the provided code |
| Abstraction | The Circle class provides an abstraction for a geometric circle. The users of this class (like the Demo class) don’t need to know the details of how the area, circumference, and diameter are calculated. They just need to know that these methods are available and what they do. | Circle class in the provided code |
| Modularity | The code is divided into separate classes with distinct responsibilities. This makes the code more modular and easier to maintain and test. Changes in one class should not affect the others. | Circle and Demo classes in the provided code |
| Readability | The code is quite readable due to the clear naming of classes, methods, and variables. This makes it easier for other developers to understand and work with the code. | Circle and Demo classes in the provided code |

1. Justify the Math.random() from the perspective of functional programming paradigm.

In functional programming, we like functions to always give the same result if we give them the same inputs. But Math.random() doesn’t do this - every time we call it, we get a different number, even though we didn’t give it any inputs. This might seem like Math.random() is breaking the rules of functional programming. But we can think of it like this: Math.random() does have an input, but it’s hidden. The input is the current state of the random number generator, which changes every time we call Math.random(). So, Math.random() is a bit unusual in functional programming terms, but we can still use it in a careful way. And it’s a good reminder that sometimes, we need to bend the rules a bit to write practical programs.

1. Improve this code snippet in Java/Javascript.

|  |
| --- |
| let PI = 3.14;  const calculateArea = (radius) => radius \* radius \* PI;  console.log( calculateArea(10) ); |

const PI = 3.14; // better precision using ‘const’ instead of ‘let’

const calculateArea = (radius) => radius \* radius \* PI;

console.log(calculateArea(10));

1. Discuss in group the impact of purity in computer programming.

Predictability: Pure functions always produce the same output for the same input, which makes the code easier to understand and predict.

Testability: Pure functions are easier to test because you only need to check the outputs for a range of inputs. There’s no need to set up and tear down state for each test.

Parallelism: Pure functions can be run in parallel because they don’t modify shared state.

Reusability and Composability: Pure functions are modular and can be reused and composed to create more complex functions.

1. Compare and contrast between function/method and higher order function/method.

A function or method is a block of code that performs a specific task. It can take inputs (parameters) and return an output (result) whereas a higher-order function is a function that can take other functions as arguments and/or return a function as its result. Higher-order functions are a key component of functional programming.

Example of function/method:

A black background with white text

Description automatically generated

Example of higher function /method:

A computer code on a black background

Description automatically generated

1. Given the username as follows. Write a functional Javascript code to find a username called “salleh”. Display the name if found, otherwise, display a null.

const username = [ "ali", "ahmad", "maria", "john", "derick", "salleh" ];

A computer screen shot of code

Description automatically generated

1. Given an array list as follows. [12,34,21,4,56,77,88,44,885,2,5,7,98,54]

Write functional Javascript codes for the following requirements.

1. Iterate and display through all elements of array list.

ii. Search and display an element, ie., 885.

iii. Double up each array element by 2 and store them in another array list.

A screen shot of a computer code

Description automatically generated