20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Suhitha K
CB.EN.U4CYS21033
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

Outline

1. Declarative Haskell

2. Declarative Haskell - Language

3. Object Oriented PHP

4. Object Oriented PHP - Language

5. Comparison and Discussions

6. Real-World Case Studies

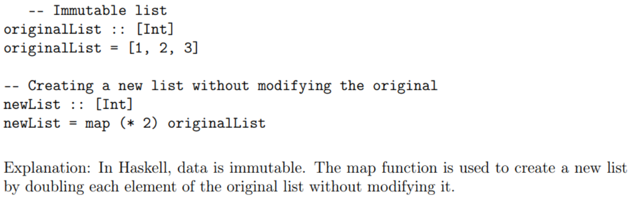
7. Bibliography

Declarative Haskell

Introduction

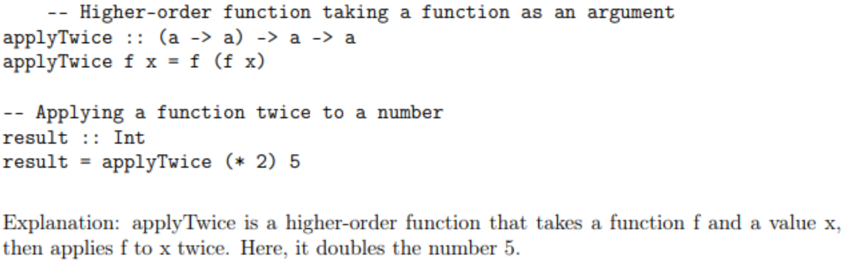
- Declarative programming paradigm focuses on expressing the "what" rather than the "how."
- Haskell, a functional programming language, exemplifies the declarative approach. **Key Characteristics of Haskell**
- Functional Programming Principles: Code is composed of pure functions without side effects.
- Immutability and Lazy Evaluation: Data is immutable, and computations are delayed until needed.
- Strong and Expressive Type System: Type inference and static typing enhance safety.
- Concurrency Support: Haskell facilitates concurrent and parallel programming.

Declarative Haskell - Language



```
   -- Immutable list
originalList :: [Int]
originalList = [1, 2, 3]

-- Creating a new list without modifying the original
newList :: [Int]
newList = map (* 2) originalList
```

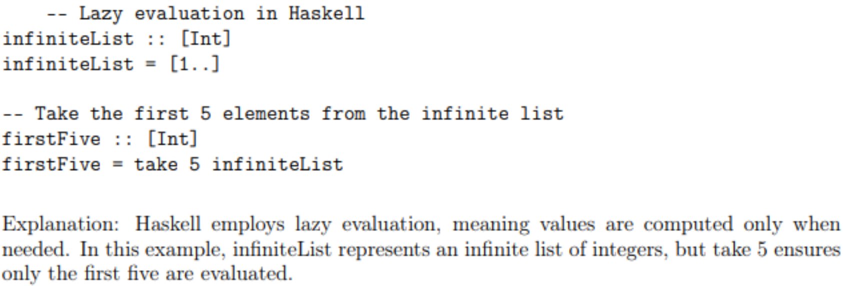
Explanation: In Haskell, data is immutable. The map function is used to create a new list by doubling each element of the original list without modifying it.



```
    -- Higher-order function taking a function as an argument
applyTwice :: (a -> a) -> a -> a
applyTwice f x = f (f x)

-- Applying a function twice to a number
result :: Int
result = applyTwice (* 2) 5
```

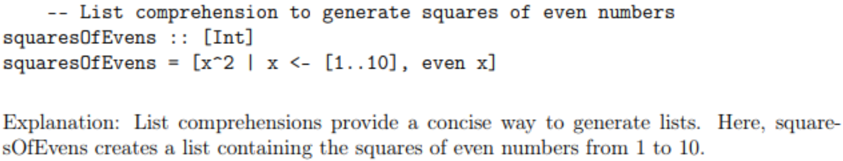
Explanation: applyTwice is a higher-order function that takes a function f and a value x, then applies f to x twice. Here, it doubles the number 5.

2



```
    -- Lazy evaluation in Haskell
infiniteList :: [Int]
infiniteList = [1..]

-- Take the first 5 elements from the infinite list
firstFive :: [Int]
firstFive = take 5 infiniteList
```

Explanation: Haskell employs lazy evaluation, meaning values are computed only when needed. In this example, infiniteList represents an infinite list of integers, but take 5 ensures only the first five are evaluated.



```
    -- List comprehension to generate squares of even numbers
squaresOfEvens :: [Int]
squaresOfEvens = [x^2 | x <- [1..10], even x]
```

Explanation: List comprehensions provide a concise way to generate lists. Here, squaresOfEvens creates a list containing the squares of even numbers from 1 to 10.

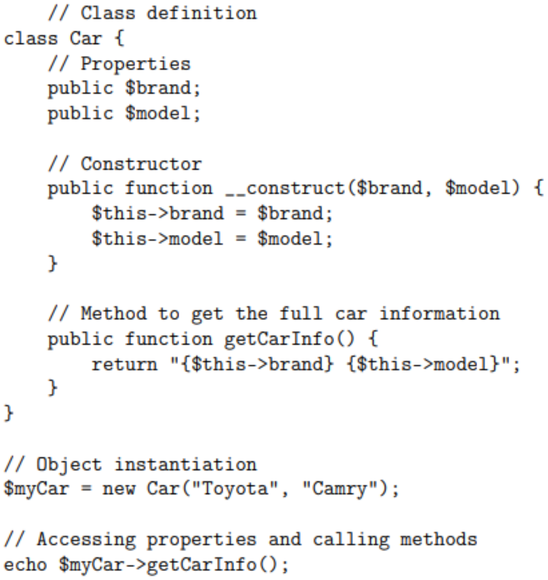
Object Oriented PHP

Introduction to Object-Oriented Programming

- Object-Oriented Principles: Encapsulation, Inheritance, Polymorphism, and Abstraction.
- PHP as an Object-Oriented Language: Leveraging classes and objects for structured code.

  Key Features of Object-Oriented PHP
- Classes, Objects, and Inheritance: Creating reusable structures and facilitating code organization.
- Encapsulation and Abstraction: Bundling data and methods within objects.
- Polymorphism and Dynamic Typing: Objects can take on multiple forms, and types can change dynamically.

Object Oriented PHP- language



```
    // Class definition
class Car {
    // Properties
    public $brand;
    public $model;

    // Constructor
    public function __construct($brand, $model) {
        $this->brand = $brand;
        $this->model = $model;
    }

    // Method to get the full car information
    public function getCarInfo() {
        return "{$this->brand} {$this->model}";
    }
}

// Object instantiation
$myCar = new Car("Toyota", "Camry");

// Accessing properties and calling methods
echo $myCar->getCarInfo();
```

Object Oriented PHP- language

Explanation: Here, we define a Car class with properties (*brandand*model), a constructor to initialize the object, and a method (getCarInfo) to retrieve car information. We then create an instance of the Car class and call its method.

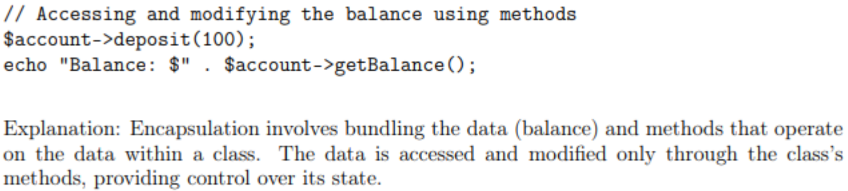
```
    // Parent class
class Animal {
    public function makeSound() {
        echo "Generic animal sound";
    }
}

// Child class inheriting from Animal
class Dog extends Animal {
    // Override makeSound method
    public function makeSound() {
        echo "Woof!";
    }
}

// Object instantiation
$dog = new Dog();

// Calling the overridden method
$dog->makeSound();
```

PHP

Explanation: Inheritance allows a class (Dog) to inherit properties and methods from another class (Animal). The Dog class overrides the makeSound method to provide a specific implementation.

```
    // Class with encapsulation
class BankAccount {
    private $balance = 0;

    // Method to get the balance
    public function getBalance() {
        return $this->balance;
    }

    // Method to deposit money
    public function deposit($amount) {
        $this->balance += $amount;
    }
}

// Object instantiation
$account = new BankAccount();
```

PHP



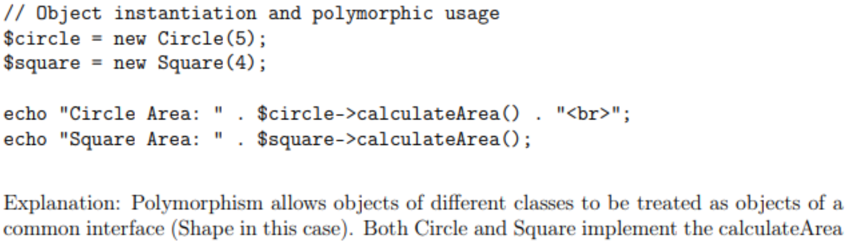
```
// Accessing and modifying the balance using methods
$account->deposit(100);
echo "Balance: $" . $account->getBalance();
```

Explanation: Encapsulation involves bundling the data (balance) and methods that operate on the data within a class. The data is accessed and modified only through the class's methods, providing control over its state.

PHP

```
    // Polymorphism with interfaces
interface Shape {
    public function calculateArea();
}

class Circle implements Shape {
    private $radius;

    public function __construct($radius) {
        $this->radius = $radius;
    }

    public function calculateArea() {
        return pi() * $this->radius * $this->radius;
    }
}

class Square implements Shape {
    private $side;

    public function __construct($side) {
        $this->side = $side;
    }

    public function calculateArea() {
        return $this->side * $this->side;
    }
}

// Object instantiation and polymorphic usage
$circle = new Circle(5);
```

PHP



```
// Object instantiation and polymorphic usage
$circle = new Circle(5);
$square = new Square(4);

echo "Circle Area: " . $circle->calculateArea() . "<br>";
echo "Square Area: " . $square->calculateArea();
```

Explanation: Polymorphism allows objects of different classes to be treated as objects of a common interface (Shape in this case). Both Circle and Square implement the calculateArea

Comparision and Discussions

Syntax and Expression Differences

- Haskell:Concise and expressive syntax.
- PHP:More flexible syntax, familiar to developers from other languages.

Execution Differences and Impact

- Haskell: Lazy evaluation can optimize memory usage.
- PHP:Eager execution, potentially impacting performance.

Advantages and Challenges

Haskell:

- Strengths in mathematical precision and functional purity.
- Challenges in a steeper learning curve.

PHP:

- Strengths in building web-based applications.
- Challenges in managing complex object interactions.

Real-World Case Studies

Declarative Haskell Applications

1. Financial Modeling:Functional data structures and immutability ideal for complex calculations.
2. Data Analysis:Lazy evaluation and higher-order functions excel in processing large datasets.
3. Compiler Design: Purity aligns well with theoretical foundations of compiler construction.

Object-Oriented PHP Applications

1. Content Management Systems (CMS): Object-oriented structure for building dynamic and interactive websites.
2. E-commerce Platforms:Inheritance and polymorphism simplify complex shopping cart systems.
3. Social Networking Sites: Modeling user interactions and relationships using OOP principles.

References

- https://wiki.haskell.org/AbriefintroductiontoHaskell
- https://www.quora.com/Can-Haskell-be-considered-a-declarative-programming-language
- https://wiki.haskell.org/Introduction
- web.engr.oregonstate.edu/ erwig/papers/DeclScriptingSLE09.pdf
- https://www.quora.com/Can-Haskell-be-considered-a-declarative-programming-language
- http://wiki.haskell.org/AbriefintroductiontoHaskell
- https://www.haskell.org/
- https://wiki.haskell.org/Books
- https://www.reddit.com/r/learnprogramming/comments/114dyd2/programmingshifted
- https://stackoverflow.blog/2020/09/02/if-everyone-hates-it-why-is-oop-still-so-widely-spread/
- https://www.techtarget.com/searchapparchitecture/tip/The-basics-of-working-with-declarative-programming-languages
- https://programiz.pro/resources/imperative-vs-declarative-programming/
- https://www.php.net/manual/en/language.oop5.php
- https://www.w3schools.com/php/phpoopintro.asp
- https://www.geeksforgeeks.org/php-classes-and-objects/
- https://www.tutorialspoint.com/php/phpobjectoriented.html