20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by G.Hemesh

CB.EN.U4CYS21020

TIFAC-CORE in Cyber Security

Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Object oriented paradigm

- Object Oriented Paradigm,also known in short as OOP is defined as programming paradigm which is built on the concept of object and classes.
- It is employed to organise a software application into easily reusable, basic code blueprints known as classes that are utilised to generate unique instances of objects.
- Object-Oriented Paradigm approach identifies classes of objects that are closely related to the methods with which they are associated. It also covers the concepts of attribute and method inheritance.
- Class is a user-defined data type that can be accessed and used by creating an instance of that class. It has its own data members and member functions. A class is comparable to an object's blueprint
- An Object refers to an instance of a class, which is a blueprint or template for creating objects.They are the fundamental building blocks of OOP which is used to model real-world entities.
- Advantages of OOP: 1. Increase productivity in software development 2. Makes troubleshooting simpler 3. Reinforces security 4. Provides design advantages 5.Lowers development costs. 6. Code Flexibility through polymorphism

Features of OOP

- Object Oriented Paradigm is implemented with a set of key concepts and features:
- Classes are templates that define the structure and behaviour of objects while Objects are instances of classes created at runtime and has its own set of attributes.
- Encapsulation is bundling of data and methods that operates on data within aclass/object and helps in hiding internal details of an object
- Abstraction allows the programmer to focus on essential features and hiding the irrelevant details. It creates abstract classes and interfaces which contain attributes and methods without implementation.
- Inheritance refers to the process of gaining properties where derived class can inherit from base class. It allows code reuse and provides parent-child relation.
- Polymorphism enables a single interface to represent various type of objects,achieved through method overriding. Compile Time Polymorphism and Runtime Polymorphism are two different types of polymorphism.

Example of OOP in common Languages

```cpp
#include <iostream>
#include <string>

class Person {
private:
    std::string name;
    int age;

public:
    Person(const std::string& n, int a) : name(n), age(a) {}

    void displayInfo() {
        std::cout << "Name: " << name << ", Age: " << age << " years ol
    }
};

int main() {
    Person person1("John Doe", 25);
    person1.displayInfo();

    return 0;
}
```

- Implementation of OOP Concepts such as Abstraction, Inheritance and Polymorphism in common languages i.e. Python in the above code

Intruduction to kotlin

- Kotlin: A Modern, Statically Typed Language
- Modern, Concise Syntax: Kotlin is a modern programming language that maintains compatibility with Java while offering a more concise and expressive syntax. It eliminates boilerplate code, making it more readable and reducing the likelihood of errors. The language prioritizes developer productivity and code clarity
- Full Java Interoperability:Kotlin seamlessly interoperates with Java, allowing developers to leverage existing Java libraries and frameworks. This makes Kotlin an excellent choice for projects with a Java codebase, enabling a smooth transition and integration of Kotlin features.
- Null Safety:Kotlin places a strong emphasis on null safety, addressing the common issue of null pointer exceptions. The type system distinguishes between nullable and non-nullable types, enhancing code reliability by catching potential null-related errors at compile-time.

Intruduction of kotlin

- Coroutines for Asynchronous Programming: Kotlin introduces coroutines, providing a concise and expressive way to handle asynchronous programming. Coroutines simplify the complexity of asynchronous tasks, offering a more readable and sequential coding style without resorting to traditional callback patterns.
- Smart Casts and Extension Functions:Kotlin includes features like smart casts, which automatically handle type casting, reducing the need for explicit casting. Extension functions enable developers to augment existing classes with new functionality, promoting modularity and extensibility in the code.

Advantages of kotlin

- Interoperability with Java:Kotlin seamlessly integrates with Java, allowing developers to take advantage of existing Java codebases and libraries.
- Null Safety:Kotlin's strong emphasis on null safety helps prevent null pointer exceptions and enhances the reliability of code.
- Concise Syntax:Kotlin's concise syntax reduces boilerplate code, making the language more readable and improving developer productivity.
- Coroutines for Asynchronous Programming:Kotlin's coroutines provide an efficient and readable way to handle asynchronous tasks, simplifying complex asynchronous code.
- Smart Casts and Extension Functions:Features like smart casts and extension functions contribute to more expressive, modular, and adaptable code.

kotlin basic code

```kotlin
fun greet(name: String) = "Hello, $name!"

fun main() {
    println("Welcome to Kotlin!")

    val age = 25
    var isStudent = true

    println(if (isStudent) "You are a student." else "You are not a st

    println(greet("Alice"))

    for (i in 1..5) println("Count: $i")

    listOf("Apple", "Banana", "Orange").forEach { println("Fruit: $it")

    val nullableName: String? = null
    println("Length of name: ${nullableName?.length}")

    val day = "Monday"
    val typeOfDay = when (day) {
        "Monday", "Tuesday" -> "Weekday"
        "Saturday", "Sunday" -> "Weekend"
        else -> "Unknown"
    }

    println("Type of day: $typeOfDay")
```

Figure: Kotlin code

Intruduction to Logic - MiniZinc

- MiniZinc is a special language for solving complex problems using constraints. It lets you describe the rules and goals of a problem without worrying about exactly how to solve it. This makes it easier to focus on the problem's logic rather than the nitty-gritty details of the solution. In short, it's a tool for expressing and solving tricky problems in a straightforward way.
- Declarative Modeling Language:MiniZinc is a high-level language for expressing combinatorial problems using a declarative syntax.
- Constraint Programming:It focuses on constraint programming, allowing users to define logical constraints and relationships between variables.
- Natural Language-Like Syntax:The language uses a syntax that closely resembles natural language, making it accessible to both programmers and domain experts.

Intruduction to Logic - Minizinc

- Solvers Integration:MiniZinc is not a solver itself; it interfaces with solvers like Gecode or Chuffed to find solutions or optimize according to specified criteria.
- Variables and Constraints:Users declare variables to represent unknowns and specify constraints to define logical relationships, guiding the solver to find solutions.
- Problem Solving:MiniZinc supports both satisfaction problems (finding any solution) and optimization problems (finding the best solution).
- Example - N-Queens:As an example, the N-Queens problem can be expressed in MiniZinc by declaring variables and specifying constraints to ensure queens do not threaten each other on a chessboard.

Advantages of Minizinc

- Solver Compatibility:It supports various solvers, allowing users to choose the best solver for their specific problem.
- Expressive Language:The language is concise and expressive, enabling clear representation of complex problems.
- Community Support:An active community provides support, tutorials, and a repository of models for various problems.
- Open Source:Being open source, MiniZinc allows user contributions and ensures ongoing development.
- Parallel Execution:Some solvers support parallel execution, enhancing solving times for certain problems.

Minizinc basic code

```
% Define the size of the chessboard
int: N;

% Declare variables for the queens' positions
array[1..N] of var 1..N: queens;

% Constraints to ensure queens do not threaten each other
constraint all_different(queens) /\
           all_different([queens[i]+i | i in 1..N]) /\
           all_different([queens[i]-i | i in 1..N]);

% Output the solution
output ["\(queens)"];
```

Figure: Minizic code

comparison

- Kotlin and MiniZinc serve different purposes. Kotlin is a versatile programming language for building applications, while MiniZinc is a specialized language for expressing and solving combinatorial problems using constraint programming. The choice between them depends on the nature of the problem and the requirements of the application or domain.

Real world example of kotlin

- Android App Development:
  Description: Kotlin is officially supported as a first-class language for Android development. Many Android applications, including those developed by major companies like Pinterest, Trello, and Evernote, use Kotlin extensively. Benefits: Kotlin provides concise syntax, improved null safety, and seamless interoperability with existing Java code, making it a preferred choice for Android developers.
- Data Science and Analysis:
  Description: Kotlin is increasingly being used in data science projects. Libraries like KotlinDL provide support for deep learning, and the language's versatility allows developers to work on various data-related tasks. Benefits: Kotlin's statically typed nature and modern features enhance code quality and maintainability in data science workflows.

Real world example of minizinc

- Scheduling and Timetabling:
  Description: MiniZinc is frequently employed in scheduling problems, such as employee scheduling, project scheduling, or course timetabling. It helps find optimal solutions considering constraints like resource availability, deadlines, and preferences. Benefits: MiniZinc's constraint modeling capabilities make it well-suited for efficiently solving complex scheduling problems
- CSP Competitions:
  Description: MiniZinc is a standard modeling language in Constraint Satisfaction Problem (CSP) competitions. These competitions aim to solve challenging real-world problems using constraint programming techniques. Benefits: The use of MiniZinc in CSP competitions showcases its versatility and effectiveness in solving a wide range of complex problem

bibliography

- kotlin:
- "Kotlin in Action" by Dmitry Jemerov, Svetlana Isakova
- "Programming Kotlin" by Venkat Subramaniam
- minizinc:
- "MiniZinc: Towards a Standard CP Modeling Language" by G. Nethercote and P.J. Stuckey
- "Handbook of Constraint Programming" edited by Francesca Rossi, Peter van Beek, Toby Walsh

Refrences

- https://kotlinlang.org/docs/getting-started.html
- https://www.w3schools.com/KOTLIN/index.php
- https://www.minizinc.org/
- https://github.com/MiniZinc