20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Vinoth Kumar C
CB.EN.U4CYS21085
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Programming Paradigm

1. Paradigm can also be termed as method to solve some problem or do some task
2. It reflects a set of principles, concepts, and practices that programmers follow to structure their code and solve problems.
3. Different paradigms provide distinct ways to organize and structure code, determining how data is represented and manipulated.
4. There are various programming paradigms. They are broadly divided in two sub classes. They are:
    1. Imperative programming paradigm
    2. Declarative programming paradigm

Reactive Programming Paradigm

1. Reactive programming describes a design paradigm that relies on asynchronous programming logic to handle real-time updates to otherwise static content.
2. It provides an efficient means – the use of automated data streams – to handle data updates to content whenever a user makes an inquiry.
3. Data streams used in reactive programming are coherent, cohesive collections of digital signals created on a continual or near-continual basis.
4. These data streams are sent from a source – such as a motion sensor, temperature gauge or a product inventory database – in reaction to a trigger.
5. Reactive programming and the reactive systems it deals with consist of a combination of "observer" and "handler" functions.
6. The former recognizes important conditions or changes and generates messages to signal they've happened, and the latter deals with those messages appropriately.

Reactive

Here are some key concepts and principles

1. **Asynchronous and Event-Driven:** At its core, reactive programming is designed to handle asynchronous operations and events. It allows developers to build systems that react to changes in the environment, user inputs, or data streams without blocking the execution of the program.
2. **Observables and Observers:** The fundamental building blocks of reactive programming are observables and observers. Observables represent data streams that emit values or events over time, and observers subscribe to these observables to react to the emitted data. This creates a clear separation between the producer of data and the consumer.
3. **Declarative Approach:** Reactive programming encourages a declarative programming style, where developers describe what should happen in response to events rather than specifying the step-by-step process of how it should happen. This leads to more concise and expressive code.

Reactive

Here are some key concepts and principles

1. **Data Transformation with Operators:** Reactive programming provides a rich set of operators that allow developers to transform, filter, combine, and manipulate data streams. These operators make it easy to express complex data transformations in a declarative and composable manner.
2. **Reactivity to Changes:** Reactivity in the context of reactive programming refers to the ability of a system to react dynamically to changes. This includes changes in data, changes in user input, or changes in the system's state. Reactive systems are designed to be responsive and adaptive.
3. **Error Handling and Resilience:** Reactive systems provide mechanisms for handling errors in a consistent way. This includes error handling within data streams, allowing developers to propagate and handle errors without causing the entire application to fail.

Implementation:

```
const { Observable } = require('rxjs');

// Create an observable that emits values every second
const observable = new Observable(observer => {
  let count = 1;

  const intervalId = setInterval(() => {
    observer.next(count);

    // Stop after emitting 5 values
    if (count === 5) {
      observer.complete();
      clearInterval(intervalId);
    }

    count++;
  }, 1000);
});

// Subscribe to the observable
observable.subscribe({
  next: value => console.log(`Received: ${value}`),
  complete: () => console.log('Observable completed'),
});
```

Figure: Reactive

ReactJS Programming Language

1. ReactJS, developed by Facebook, is a powerful open-source JavaScript library widely used for building user interfaces.
2. Its hallmark is a component-based architecture, allowing developers to create modular and reusable UI elements.
3. React's declarative syntax and efficient use of a Virtual DOM contribute to high-performance rendering.
4. JSX, a JavaScript extension, simplifies the integration of HTML-like code directly into JavaScript, enhancing code readability.
5. The library enforces a unidirectional data flow, ensuring a clear and maintainable structure.
6. React's state and props mechanism facilitates dynamic UIs, and the introduction of React Hooks in version 16.8 enables stateful logic in functional components.

ReactJS

Here are some key concepts and principles:

1. **Component-Based Architecture:** React follows a component-based architecture, allowing developers to build UIs by creating reusable and modular components. Components encapsulate both the UI and the logic associated with it.
2. **Virtual DOM:** React introduces a Virtual DOM, a lightweight in-memory representation of the actual DOM. This allows React to efficiently update and render only the components that have changed, reducing the performance impact of direct DOM manipulation.
3. **Unidirectional Data Flow:** React enforces a unidirectional data flow, meaning that data flows in a single direction—from parent components to child components. This helps maintain a predictable and easily understandable data flow within the application.

Here are some key concepts and principles:

1. **Reconciliation Algorithm:** React employs a reconciliation algorithm that efficiently updates the UI by comparing the Virtual DOM with the previous state and determining the minimal set of changes needed to reflect the new state.
2. **State and Props:** React components can have both state and props. State represents the internal state of a component, while props (short for properties) are inputs passed to a component, allowing it to be customizable and reusable.
3. **Lifecycle Methods:** React components go through a lifecycle, and developers can hook into various lifecycle methods to execute code at different stages, such as when a component is mounted, updated, or unmounted.

Implementation:

```jsx
import React, { useState } from 'react';

const CounterExample = () => {
  // State to store the count
  const [count, setCount] = useState(0);

  // Function to handle incrementing the count
  const handleIncrement = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <h1>Simple Counter Example</h1>
      <p>Count: {count}</p>
      <button onClick={handleIncrement}>Increment</button>
    </div>
  );
};

export default CounterExample;
```

Figure: ReactJS

Logic Programming Paradigm

1. A logic programming paradigm is a set of principles and techniques that guide the design and implementation of logic programs.
2. A logic program consists of a collection of facts and rules that describe the relationships and properties of entities, and a query language that allows asking questions and obtaining answers from the program.
3. A logic programming paradigm defines the syntax and semantics of the facts, rules, and queries, as well as the inference mechanism that derives new facts and rules from the existing ones.
4. Logic programming has several advantages over other programming paradigms, such as being declarative, expressive, and flexible.
5. Being declarative means that logic programs focus on what the program should do, rather than how it should do it, making them easier to understand, maintain, and modify.

Logic

Here are some key concepts and principles:

1. **Logical Statements:** Programs in logic programming are expressed as logical statements. These statements are often written in the form of Horn clauses, consisting of a head (conclusion) and a body (premises).
2. **Declarative Nature:** Logic programming is a declarative paradigm, meaning that programs specify what needs to be achieved rather than providing a step-by-step procedure for achieving it. Developers focus on describing relationships and rules.
3. **Predicates and Facts:** Predicates represent relationships or properties, and facts are instances of predicates. These are used to express the knowledge and information within a logic program.

Logic

Here are some key concepts and principles:

1. **Rules and Horn Clauses:** Programs are composed of rules, and the basic building blocks are often Horn clauses. A Horn clause consists of a head and a body, defining implications and conditions.Logical Inference:
2. **Logical Inference:** Execution in logic programming involves logical inference, where the system deduces conclusions based on the given logical rules and facts. The process of inference is driven by goals or queries posed to the system.
3. **Prolog:** Prolog (Programming in Logic) is a popular language associated with the Logic Programming Paradigm. It provides a framework for expressing rules and performing logical inference.

Implementation:

```
% Facts: Family relationships
parent(john, ann).
parent(john, bob).
parent(mary, ann).
parent(mary, bob).
parent(ann, charlie).

% Rules: Define relationships
father(X, Y) :- male(X), parent(X, Y).
mother(X, Y) :- female(X), parent(X, Y).
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).

% Facts: Define genders
male(john).
male(bob).
male(charlie).
female(mary).
female(ann).

% Query examples
% Is John the father of Charlie?
% Query: father(john, charlie).
% Expected result: true

% Who are the grandchildren of John?
% Query: grandparent(john, X).
% Expected result: X = charlie
```

Figure: Logic

Datalog Programming Language

1. Datalog's power extends far beyond simple queries. Its declarative nature and rich logic unlock a vast potential for complex reasoning and problem-solving.
2. Datalog isn't just about finding existing connections; it excels at inferring new knowledge. Imagine your data tells you Alice loves strawberries and strawberries are red.
3. Datalog's rules can reference themselves, creating loops that iterate and discover complex patterns.
4. Datalog isn't an island. It seamlessly integrates with various databases and programming languages, playing nicely with your existing tech stack. This opens doors to automating tasks like data analysis, knowledge extraction, and rule-based decision making. Imagine automatically generating financial reports or managing security audits – Datalog makes it possible.

Datalog

Here are some key concepts and principles:

1. **Declarative Nature:** Datalog is a declarative language, which means that users specify what they want to achieve rather than specifying how to achieve it. Queries in Datalog resemble formal logical statements.
2. **Rule-Based Programming:** Datalog programs consist of rules that define relationships and facts. These rules are used to derive new information from existing data.
3. **Database Query Language:** Datalog is primarily used as a query language for deductive databases. It allows users to express complex queries, including recursive queries and transitive closures.

Datalog

Implementation:

```
% Facts: Family relationships
parent(john, ann).
parent(john, bob).
parent(mary, ann).
parent(mary, bob).
parent(ann, charlie).

% Rules: Define relationships
father(X, Y) :- male(X), parent(X, Y).
mother(X, Y) :- female(X), parent(X, Y).
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
sibling(X, Y) :- parent(Z, X), parent(Z, Y), X != Y.

% Facts: Define genders
male(john).
male(bob).
male(charlie).
female(mary).
female(ann).

% Query examples
% Is John the father of Charlie?
% Query: father(john, charlie).

% Who are the grandchildren of John?
% Query: grandparent(john, X).

% Are Ann and Bob siblings?
% Query: sibling(ann, bob).
```

Figure: Datalog

Comparison and Discussions

Reactive Programming	**Reactive Programming**
Primarily used for building responsive and event-driven systems, such as user interfaces, real-time applications, and streaming data processing.	Mainly used for knowledge representation, reasoning, and rule-based systems. It's applied in areas like artificial intelligence, expert systems, and databases.
Focuses on the flow of data and events, often involving observables, streams, and asynchronous processing.	Focuses on expressing relationships, rules, and logical conditions. Involves the use of predicates, facts, and logical inference.
Typically doesn't explicitly handle negation; it focuses on data flows and events.	Can include explicit negation handling, allowing developers to express conditions that exclude certain facts or relationships.
Learning curve may involve understanding reactive patterns, observables, and asynchronous programming.	Learning curve may involve understanding logical inference, rule-based systems, and the use of predicates.

Comparison and Discussions

ReactJS	Datalog
Primarily used for building user interfaces, particularly in web development. ReactJS excels in creating dynamic and responsive UIs for single-page applications.	Primarily used for knowledge representation, reasoning, and querying databases. Datalog is employed in areas like artificial intelligence and expert systems for expressing logical relationships.
Focuses on abstracting the UI layer, managing state, and handling user interactions. ReactJS is designed for building interactive and dynamic user interfaces.	Focuses on abstracting logical relationships and rules. It is used for expressing queries and relationships in a database-centric context.
ReactJS is a JavaScript library for building user interfaces, following a component-based and reactive programming paradigm.	Datalog is a logic programming language, emphasizing rules, queries, and logical inference.
Manages UI state reactively, updating components based on changes in state. State changes trigger UI updates.	Manages data relationships and inference, inferring new facts based on existing rules and data.
ReactJS doesn't inherently involve querying data. It focuses on managing and rendering UI components based on state changes.	Datalog is specifically designed for querying databases. It involves expressing queries using logic rules and retrieving data based on those queries.

Bibliography

1. https://www.baeldung.com/cs/reactive-programming
2. https://www.geeksforgeeks.org/reactive-programming-in-java-with-example/
3. https://medium.com/sysco-labs/reactive-programming-in-java-8d1f5c648012/
4. https://www.youtube.com/watch?v=EExlnnq5Grs/
5. https://www.linode.com/docs/guides/logic-programming-languages/
6. https://www.youtube.com/watch?v=BfEjDD8mWYg/
7. https://react.dev
8. https://legacy.reactjs.org
9. https://www.w3schools.com/REACT/DEFAULT.ASP
10. https://en.wikipedia.org/wiki/Datalog
11. https://datalog.co.in
12. https://clojure.github.io/clojure-contrib/doc/datalog.html
13. https://www.geeksforgeeks.org/difference-between-functional-and-logical-programming/
14. https://www.codium.ai/glossary/programming-logic/
15. chatgpt