Java Review Session 6

CS 5004

Topics Covered

Java Syntax

- Primitive-type Streams
- Parallel Stream Processing
- Multithreading (very quick)
- XML

Design Pattern Fundamentals

- The essential elements of a design pattern
- The different types of design patterns
- Model View Controller (MVC)

Multithreading

Thread - a program unit that is executed concurrently with other parts of the program

The JVM executes each thread for as short period of time and then switches to another thread (this gives the illusion of them executing in parallel)

If you have multiple CPUs, then threads can actually be run in parallel (one on each processor)

Parallel Stream Processing

Demo

Extensible Markup Language (XML)

- A popular mechanism for encoding data
- Uses a mixture of text and tags to describe data
- XML allows you to encode complex data, independent of any programming language, in a form that can be easily parsed
- XML files are readable by computer and by humans
- An element is a unit of information that is delimited by a start tag and a matching end tag
    - Elements can contain other elements
- Can be used to represent a wide array of data

XML Example

An element with three child elements:

```
<address>
      <street>1195 W. Fairfield Rd.</street>
      <city>Sunnyvale</city>
      <state>CA</state>
</address>
```

Plain text:

```
Toaster

Price
```

Design Patterns

Design Patterns

What is a design pattern?

- A reusable solution to a common problem within a given context in software design
- They are sort of like templates to help us write code
  - Provide structure and organization for code used to solve a particular problem (they describe object-oriented designs)
  - Descriptions of objects and classes that can be used to solve a general design problem in a particular context

What do they do?

- A design pattern names, abstracts, and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design

Essential elements of a design pattern

The four main elements:

1. **Pattern name** - a meaningful and concise way to identify the pattern
   - Helps us to discuss and communicate the pattern effectively
2. **Problem** - the problem and context where the pattern is applicable
3. **Solution** - the general arrangement of objects or classes to solve the problem (a general blueprint or abstract description of how to solve it)
4. **Consequences** - the implications of choosing to use a particular design pattern tradeoffs (could be in terms of pros/cons, flexibility, performance, scalability, and potential side effects of using a particular pattern)

Types of design patterns

There are three major types of design patterns:

1. **Creational Patterns** - deal with object creation mechanisms (creating objects in a way that is suitable for a particular situation)

2. **Structural Patterns** - deal with object composition or the arrangement of classes and objects to form larger structures

3. **Behavioral Patterns** - deal with communication between objects, how they interact, and how responsibilities are distributed among them

Design Patterns

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method (107)	Adapter (class) (139)	Interpreter (243) Template Method (325)
	Object	Abstract Factory (87) Builder (97) Prototype (117) Singleton (127)	Adapter (object) (139) Bridge (151) Composite (163) Decorator (175) Facade (185) Flyweight (195) Proxy (207)	Chain of Responsibility (223) Command (233) Iterator (257) Mediator (273) Memento (283) Observer (293) State (305) Strategy (315) Visitor (331)

Image: Design Patterns: Elements of Reusable Object-Oriented Software

Model View Controller (MVC)

- MVC is an architectural design pattern that is comprised of three different interconnected components:
  - The Model
  - The View
  - The Controller
- Often used for developing user interfaces
- Benefits of MVC?
  - Separation of concerns (each component has its own responsibility)
  - Overall usability/maintainability

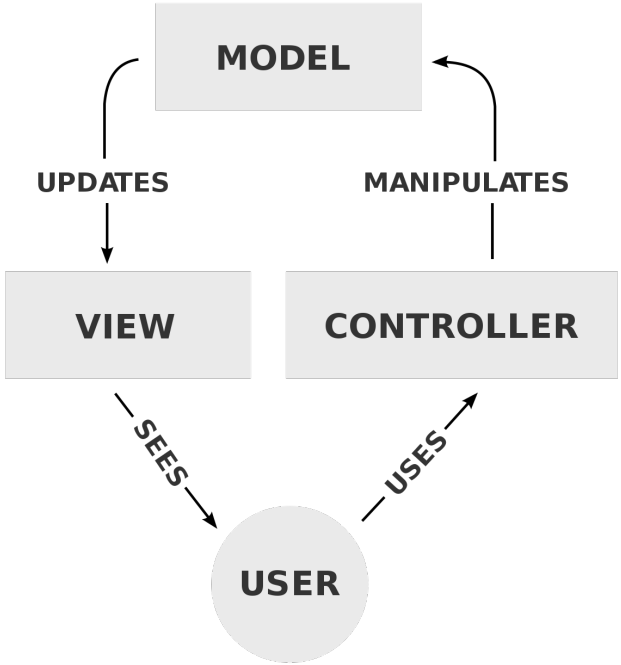

Image: Wikipedia

Model

- Represents the data and the “business logic” of the application
  - Business logic is the set of rules of procedures that dictate how data is created, stored, and manipulated
- Directly manages the data, logic, and rules of the application
  - Enforces these rules to make sure that they are followed
- May retrieve data from a database

View

- Represents the User Interface (UI) of the application
- Displays data to the user and sends commands to the controller
- Gets its data from the model and renders/displays it to the user
    - Does not contain any business logic!

- Note: will connect this to GUIs later!

Controller

- Acts as the intermediary between the model and the view
- Receives input from the user from the view, processes it (possibly changing the state of the model), and then returns the output display to the view
- Handles the user input and updates the model and the view accordingly