20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Vinoth Kumar D
CB.EN.U4CYS21086
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Object-Oriented Programming (OOP)

→ Organizes software design around data or object
→ Implemented or (Well suited) for large, complex and actively updated or maintained
→ Code reusability, scalability and efficiency.
→ It uses data Modeling - process of creating a simplified diagram of a software system.

Structure:

1. Classes
2. Objects
3. Methods
4. Attributes

Will see with an example with our Human beings:

1. Classes → Human
2. Objects → Name
3. Properties → Email, Address
4. Methods → Verify, send mail

OOPs continued

Main principle:

1. Encapsulation:

   A way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object.

2. Abstraction:

   Process of generalizing concrete details, such as attributes, away from the study of objects and systems to focus attention on details of greater importance.

3. Inheritance:

   In basic financial term describing the assets passed down to individuals after someone dies.

   Inheritance is the mechanism of basing an object or class upon another object or class, retaining similar implementation.

4. Polymorphism:

   The ability of a message to be displayed in more than one form.

OOPs continued

Implementation:

- Application development
- Game development
- Web development
- Operating systems $\rightarrow$ like Microsoft Windows uses OOPs Principle
- Databases $\rightarrow$ ObjectDB and db40

Popular Languages:

Java, Cpp, Python, Ruby

Ruby (OOPS)

→ Ruby is a dynamic, object-oriented programming language.
→ Doesn't talk directly to hardware
Instead:

1. written in a textfile
2. parsed on the interpreter
3. turned into code

→ It is a highly portable general purpose language
→ Great for

1. Desktop Application
2. Static Websites
    1. Web servers
    2. Dockers
    3. Web scrapping
    4. Crawling
3. Data processing services
4. Automation tools

Ruby continued

→ **Classes and Objects**

- classes are used to define blueprints for objects.

→ **Inheritance**

- Ruby supports single inheritance.
- '<' symbol is used.

→ **Encapsulation**

- Supports encapsulation through access specifier - Public, Private and Protected.
- Public methods can be called from outside the class and Private methods are only accessible within the class.

→ **Polymorphism**

- Allowing objects of different classes to be treated as objects of a common superclass.

Aspect-Oriented Programming (AOP)

First aspect means a particular part or feature of something.
→ It can be defined as the breaking of code into different modules
→ Aspects enable the implementation of crosscutting concerns such as- transaction, logging not central to business logic without cluttering the code core to its functionality
Dominant Frameworks in AOP:

1. AspectJ
2. JBoss
3. Spring

Main Principle:
Aspect-Oriented Programming is the separation of concerns this is the main principle.

AOP continued

Implementation:
- Aspects encapsulate cross-cutting concerns. They define what code needs to be executed and where it should be applied in the program.
- Pointcuts specify the join points in the program where the aspects should be applied.

Structure:

1. Core Modules
2. Aspects
3. Pointcuts
4. Advice
5. Weaver

JBoss (AOP)

→ JBoss AOP is a 100 percent Pure Java Aspected Oriented Framework
→ JBoss AOP is not only a framework, but also a prepackaged set of aspects that are applied via annotations, pointcut expressions, or dynamically at runtime. Some of these include caching, asynchronous communication, transactions, security, remoting, and many more.
→ For example, metrics is one common aspect. To generate useful logs from your application, you have to (often liberally) sprinkle informative messages throughout your code.

JBoss continued

Some key terms

1. *Joinpoint* → A joinpoint is any point in your Java program.

→ The call of a method, the execution of a constructor, the access of a field; all these are joinpoints.

→ You could also think of a joinpoint as a particular Java event, where an event is a method call, constructor call, field access, etc.

2. *Invocation*

→ An invocation is a JBoss AOP class that encapsulates what a joinpoint is at runtime

3. *Advice*

→ An advice is a method that is called when a particular joinpoint is executed, such as the behavior that is triggered when a method is called.Another analogy is that an advice is an "event handler".

4. *Pointcut*

→ Pointcuts are AOP's expression language.

→ Just as a regular expression matches strings, a pointcut expression matches a particular joinpoint.

JBoss continued

5. Introduction

→ An introduction modifies the type and structure of a Java class.

→ It can be used to force an existing class to implement an interface or to add an annotation to anything.

6. Aspect

→ An aspect is a plain Java class that encapsulates any number of advices, pointcut definitions, mixins, or any other JBoss AOP construct.

7. Interceptor

→ An interceptor is an aspect with only one advice, named ‘invoke‘.

→ It is a specific interface that you can implement if you want your code to be checked by forcing your class to implement an interface.

→ It also will be portable and can be reused in other JBoss environments like EJBs and JMX MBeans.

Comparison and Discussions

Relationship between OOP and AOP:

- AOP is often seen as a complementary approach to OOP, not a replacement.
- It addresses some of the limitations of OOP in handling cross-cutting concerns.
- They can be used together to create more modular, maintainable, and extensible software systems.

References

https://www.trustradius.com/reviews/red-hat-jboss-enterprise-application-platform-2019-03-15-11-35-09
https://www.baeldung.com/spring-aop-vs-aspectj
https://access.redhat.com/documentation/en-us/jboss_enterprise_application_platform/5/html/administration_and_configuration_guide/jboss_aop
https://www.iaeng.org/publication/WCECS2019/WCECS2019_p118 – 123.pdf
https://chat.openai.com/share/da3d7a43-8423-4afe-a0b5-788eb1da917c
https://chat.openai.com/share/8e3209bb-1f48-4905-8960-ea413923814e
https://chat.openai.com/share/92b3f520-6cef-4591-873e-3080414ce1f9
https://arasopraza.medium.com/introduction-to-ruby-programming-language-7014c1a651d8