Week05 Reasearch

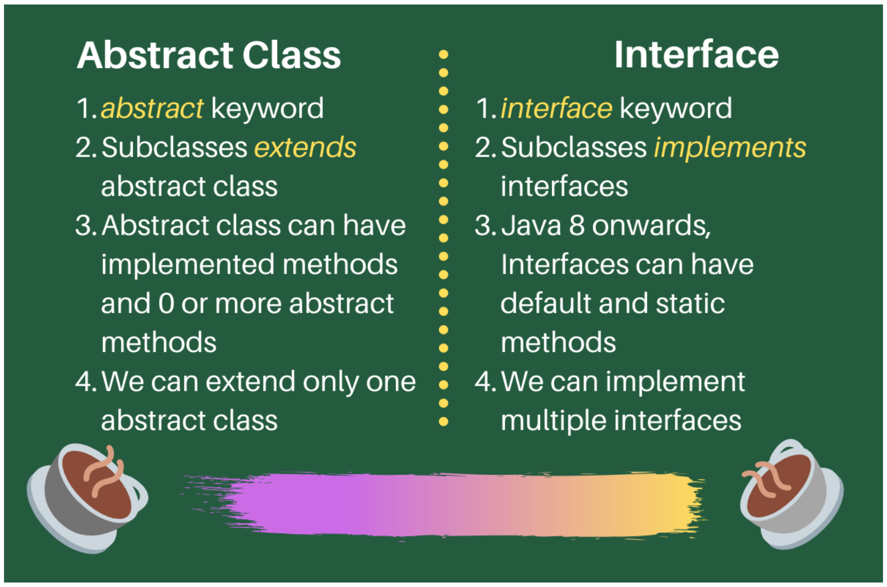
Prompts

OOP:

1. What are the four pillars of Object-Oriented Programming? Explain each pillar.
   - **Encapsulation:** It allows methods to protect data within the class or package. Data hiding.
   - **Inheritance:** It allows other child classes to have functionality from the parent class. Keeping your code DRY.
   - **Polymorphism:** The way an object can respond differently to the same function
   - **Abstraction:** Only showing what needs to be show while keeping the complexities hidden.
2. What is the relationship between a Class and an Object?
   - A **Class** is a blueprint, in which you create your properties and functionality., while an **Object** is the actual product developed from the blueprint.

Advanced OOP:

3. What are the differences between abstract classes and interfaces? When should you use one over the other?



Abstract Class	Interface
1. *abstract* keyword	1. *interface* keyword
2. Subclasses *extends* abstract class	2. Subclasses *implements* interfaces
3. Abstract class can have implemented methods and 0 or more abstract methods	3. Java 8 onwards, Interfaces can have default and static methods
4. We can extend only one abstract class	4. We can implement multiple interfaces

- The biggest difference is that we abstract classes be extended once while an interface can be implemented multiple times.

- Abstract classes should be used for objects that are closely related and interfaces should be use to provide a common functionality to unrelated classes.

4. Research the SOLID principles of Object-Oriented Programming (OOP) as introduced by Robert Martin. List the principles, and give a description of each one.

- The SOLID principles are Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation and Dependency Inversion.
    - **Single Responsibility:** a class should only have one responsibility. It will be easier for testing since it will have fewer test cases, it will have less dependencies and will be better for organization.
    - **Open/Closed:** Open for Extension, closed for modification. By doing this we stop ourselves from modifying old code and causing new potential issues.
    - **Liskov Substitution:** Being able to substitute a subtype of a class without disrupting the behavior of the program.
    - **Interface Segregation:** This simply means separating large interfaces into a few smaller ones. This ensures that implementing classes only need to be concerned about the methods that are of interest to them.
    - **Dependency Inversion:** This is referring to the decoupling of software modules. Removing the dependency of high-level modules from low-level modules, instead both will depend on abstractions.

Exceptions:

5. What is an exception?

- It is an event that occurs during the execution of a program, that disrupts the normal flow of the program. When an error occurs within a method, the method creates an object and passes it on to the runtime system.

6. What are the differences between checked and unchecked exceptions?

- The difference is that a checked exception is caught at compile time where a unchecked exception caught at runtime hence the other name "runtime exception". The checked exceptions must be handled either by re-throwing or with a catch block, whereas an unchecked exception isn't required to be handled.

Testing:

7. What is unit testing and why is it important?

- Unit testing is a software development process in which the smallest testable parts of a n application, called units, are individually tested for proper operation. This is very important because this is how you can safely provide good quality code and assure the customer that you are providing good working software.

References:

- https://learn.promineotech.com/mod/book/view.php?id=12336
- https://www.baeldung.com/solid-principles
- https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html#:~:text=Definition%3A%20An%20exception%20is%20an,off%20to%20the%20runtime%20system.