accenture
Latvia ATC

Test Automation Engineering Fundamentals: Java

Module 7: Encapsulation

Module Objectives

accenture
Latvia ATC

- At the end of this module, participants will be able to:
  - Describe the OOP principle of Encapsulation.
  - Explain the use of access modifiers in support of encapsulation.
  - Demonstrate common class design considerations supporting encapsulation.

Defining Encapsulation

accenture
Latvia ATC

- Encapsulation is the binding and hiding of data – implementing it as a single entity (a class) and wrapping it with an exposed interface.
- Other entities and objects know an instance of a class through its exposed interface, and are not concerned with its implementation.
- Encapsulation reduces the complexity of a system by separating the concerns of an object's interface from its implementation.

Access Control

accenture
Latvia ATC

- An encapsulated class should not expose details of its implementation to other objects.
- An outside object should not need to know about details of an object’s implementation in order to send messages to the object.
- These public interfaces must make sure that modifications made to an object’s state adheres to the intended design of that object’s class.

Access Modifiers

accenture
Latvia ATC

1. **Class Access** modifiers – describes how a class can be accessed

Modifier	Description
(no modifier)	Class can only be accessed from same package
public	Class can be accessed from anywhere

2. **Member Access** modifiers – describes how a member can be accessed

Modifier	Description
(no modifier)	Member is accessible within its package only
public	Member is accessible from any class of any package
protected	Member is accessible in its class package and by its subclasses
private	Member is accessible only from its class

Member Access Modifiers Diagram

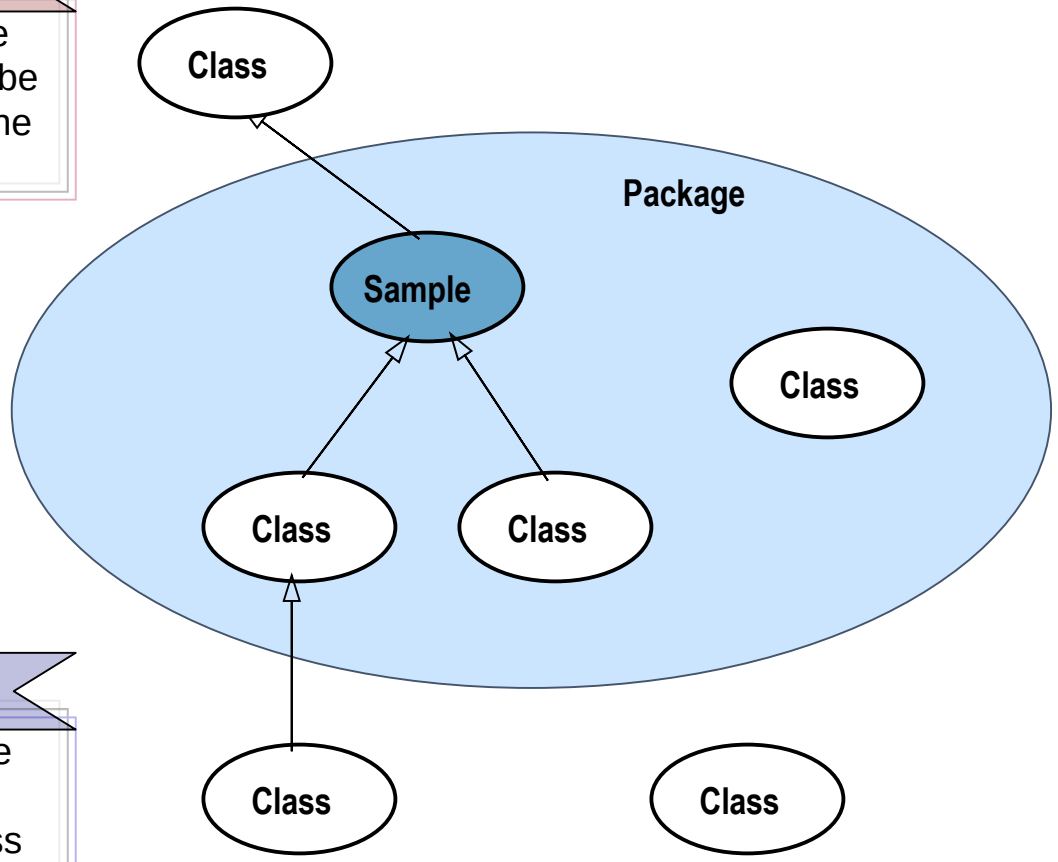
accenture
Latvia ATC

private

Private features of the Sample class can only be accessed from within the class itself.

default

Only classes that are in the package may access default features of classes that are in the package



protected

Classes that are in the package and all its subclasses may access protected features of the Sample class.

public

All classes may access public features of the Sample class.

* *Default* is not a modifier; it is just the name of the access level if no access modifier is specified.

‘Setter’ Methods

accenture
Latvia ATC

- The fields or state of an object are usually implemented by *private* attributes.
- In order to modify private attributes, objects present *public* interfaces called ‘setter’ or ‘mutator’ methods.
- Not all private attributes require setter methods; this should depend on design.
- These methods should strictly control how the fields are modified and perform appropriate validations on parameters passed.
- Any time an object’s attributes are modified, the object should validate the correctness of its state (invariants).

‘Getter’ Methods

accenture
Latvia ATC

- ‘Getter’ or ‘accessor’ methods allow objects to return the values of its private attributes.
- Not all private attributes need to have getter methods; this depends on design.
- Getter methods should always return just a copy of the values of the attributes, and not the attributes themselves.

Refer to the SetterGetterSample.java, PersonSampleOne.java and PersonSampleTwo.java sample codes.

Design According to Interfaces

accenture
Latvia ATC

- An object is known to other objects from an ‘outside’ view or through its public interfaces.
- View the design of an object from the ‘outside’ before worrying about the details of its implementation.
- By designing public interfaces, you first define the *contract* on how the instances of the class is supposed to be used and the validations that must be performed.

Refer to StrategySample.java and interface its references StrategyImplOne.java, StrategyImplTwo.java, Strategy.java and General.java sample codes.

Inheritance and Encapsulation

accenture
Latvia ATC

- Inheritance creates a dependency between the parent class and the subclass that might compromise the encapsulation of a subclass.
- Classes that can be safely inherited need extra work in order to be encapsulated properly.
  - Constructors must not call overridable methods.
  - If a method depends on a superclass method (using super.xxx()), it must be stated explicitly in it's design.
  - Interface methods as much as possible should be declared *final*, unless designed to be overridable.

Composition vs. Inheritance

accenture
Latvia ATC

- Composition is a safer alternative with regards to encapsulation when defining variations and specification in a class.
- Design variations in the behavior of a class by forwarding calls to a delegate object field and use virtual method invocation.
- Composition is more flexible since the delegate object can be determined during run-time.

Checkpoint Question

accenture
Latvia ATC

True or False?

1.Encapsulation separates the concern of an object's interface with its implementation. This, however, increases the complexity.

2.Design by Interface suggests that an object views other objects from an 'outside' view or through its public interfaces.

3.It is always a good practice to access or mutate private and public attributes of class using getters and setters.

4.Encapsulation refers to binding of data (using classes) and hiding it from the outside world (by implementing interfaces).

5.Public access modifier allows members to be accessible in its class package and by its subclasses.

Checkpoint Answers

accenture
Latvia ATC

True or False?

1.Encapsulation separates the concern of an object's interface with its implementation. This, however, increases the complexity. **False**

2.Design by Interface suggests that an object views other objects from an 'outside' view or through its public interfaces. **True**

3.It is always a good practice to access or mutate private and public attributes of class using getters and setters. **False**

4.Encapsulation refers to binding of data (using classes) and hiding it from the outside world (by implementing interfaces). **True**

5.Public access modifier allows members to be accessible in its class package and by its subclasses. **False**

Questions and Comments

accenture
Latvia ATC

- What questions or comments do you have?