Day 28 - 29th Aug 2025

Plz refer doc Day 27

Till Task 23

9.45 to 9.55 Revision

23 pax done out of 34 Pax

Sorting strategy Qn :

10.21 to 10.41

Shirisha perapagu, Sushma Mainampati, Durga bhavani Shared and expalied the code – good job

This task is from Day 26:

Task 16:

Singleton design pattern.:

Class name DManager

Add , remove , retrieve.. Methods (list)

Declare private static variable ..- type Dmanager - make sure it holds singleton instance

→ create private constructor to prevent direct instantiation of class

    → create instance..

    → throw an illegalStaticException , create **singleton** instance if not created .. use getInstance()

→ public static synchronized method – getInstance()

→ return existing instance..

→ create a new instance if doesnot exist..

In the thread safe list management:

    → declare a private variable

    → initialize the list inside the constructor using new..

    → also implement public sync methods

      → addItem

      → removeitem

      → list

Input:

Apple

Banana

Citus

D…

Done

Citus

OutPut:

Apple

Banana

D…

Task 3:

Class A {

psvm() {

Int a = 5;

Int b = 10;

Int c = 15:

sout((a > b ) && (b < c))

}

}

1. Compilation error
2. True
3. False
4. Runtime error

Task 4:

Finding Inheritance during requirement analysis is it important in OOAD .. why so?

1. It removes the need for encapsulation in the system design
2. It helps identify objects with the shared behavior to promote code reuse and logical hierarchy
3. It forces a flat class design improving performance by reducing polymorphic calls
4. Ensures all classes are instantiated using interfaces.

Task 5:

Which characteristics best defines polymorphism in OOP?

1. Ensures each class has its own copy of data members
2. It restricts method access to specific roles within a system
3. It allows a single function or operator to behave differently based on its parameters or calling object
4. It serialized different objects into a common file format for persistence

Task 6:

Which of the following best explains the concept of data hiding in Object-Oriented Programming?

1. Data hiding means removing data from memory when no longer in use to ensure memory efficiency.
2. Data hiding involves using access specifiers to restrict direct access to class members, enabling controlled interaction through methods.
3. Data hiding refers to storing object data in secure databases during runtime.
4. Data hiding is achieved by deleting unused attributes from objects after object creation.

Task 7:

In OOAD, what is the primary value of Requirements Analysis?

1. It helps define class inheritance structure before testing

2. It identifies system behavior and user needs to model objects and interactions meaningfully

3. It configures application deployment scripts for testing

4. It automatically generates interface documentation from class files

Task 8:

Which design pattern is implemented in the following code snippet?

public class ClassName {

private static ClassName instance;

private ClassName() {

public static ClassName getinstance() ( if (instance = null) {

instance = new ClassName();

}

return instance;

}

}

1. Factory Method

2. Singleton

3. Prototype

4. Builder

Task 9:

Why is Interface preferred in Java when applying polymorphism over using abstract classes in many designs?

1. Interfaces enforce tight coupling between child and parent classes

2. Interfaces offer default constructors and static fields, which abstract classes cannot

3. Interfaces allow a class to inherit from multiple sources of behavior, promoting decoupling and flexibility

4. Interfaces provide direct access to private implementation logic

Task 10:

What is the role of the "Inception Phase in the Rational Unified Process?

1. It is the final phase where deployment and user training occur.

2. It defines the runtime environment for executing object oriented code

3. It helps establish the business case, scope and feasibility of the proposed systems

4. It focuses exclusively on UI design and database integration

Task 11:

What aspect of UML Diagrams makes them crucial in Object-Oriented Analysis and Design?

1. They provide detailed flowcharts for programming logic.

2. They represent runtime logs for system monitoring purposes.

3. They visually capture the structure and behavior of systems through elements like classes, objects, and interactions.

4. They replace testing frameworks by automatically generating code

Task 12:

Why is refactoring considered a continuous part of modern software development?

1. Refactoring is performed only at the end of a release cycle for documentation purposes

2. It replaces traditional debugging with automatic patching mechanisms

3. Continuous refactoring ensures that the design evolves with changing requirements, reducing technical debt and improving code health

4. Refactoring removes dependencies to minimize source control conflicts

Task 13:

OOAD, why is the Elaboration Phase important?

1. focuses on preparing production deployment pipelines

2. is where the major architectural decisions are validated through executable prototypes and risk mitigation

3. is mainly used to finalize Uf designs and wireframes

4. is dedicated to refactoring legacy code to newer patterns

Task 14:

How are Active Objects represented in object modeling

1. As static utility classes for database access

2. As objects that encapsulates encapsulate their own threat of control and asynchronously handle requests

3. As serialized containers passed between processes

4. As Java Beans used solely for UI binding

Task 15:

What makes Composite pattern useful when designing complex tree structures?

1. It replaces the use of collections to store children

2. allows treating individual objects and compositions uniformly through a common interface.

3. It automatically serializes tree objects for persistence

4. optimizes memory by removing duplicate nodes in the tree

Task 16:

**Which** design pattern is being used in the following code snippet?

public interface ABC (

}

int doOperation(int num1, int num2);

public class OperationAdd implements ABC {

@Override

public int doOperation(int num1, int num2) (

return num1 + num2;

<

}

}

public class OperationSubtract implements ABC (

@Override

public int doOperation(int num1, int num2) { return num1- num2;

e:37.46 Mbps

}

}

public class Context (

private ABC abc;

public Context(ABC abc) {

this.abc = abc;

46 Mbps

}

public int executeABC(int num1, int num2) {

return abc.doOperation(num1, num2):

………

// more code ..