

COMP 1406Z Introduction to Computer Science II

Study Session Questions

December 26, 2025

Part 1: Principles of OOP

1. Which of the following are OOP principles? Select all that apply.
 - compilation
 - encapsulation
 - polymorphism
 - abstraction
 - inheritance
 - debugging
2. What is abstraction in Java?
 - A. hiding the implementation details and showing only the functionality
 - B. hiding the object state
 - C. writing multiple methods with the same name
 - D. using interfaces and classes
3. What must a class do if it implements an interface?
 - A. be marked as abstract
 - B. implement all abstract methods of the interface
 - C. explicitly extend the interface
 - D. contain only static methods
4. What is the main purpose of encapsulation in Java?
 - A. To restrict direct access to object data and ensure controlled access.
 - B. To provide methods to manipulate data directly.
 - C. To use runtime polymorphism.
 - D. To inherit properties from a parent class.

5.

6. What is the concept of inheritance in Java?

- A. A method of implementing interfaces for improved organization.
 - B. A process for defining private methods and variables.
 - C. A feature that ensures code executes in a specific order.
 - D. A mechanism for one class to acquire the properties and methods of another.
-
-
-

7.

8.

9. What is the difference between an abstract class and an interface in Java?

- A. an interface shouldn't have attributes or constants, an abstract class can
- B. an abstract class supports multiple inheritance, an interface does not
- C. abstract classes can have methods with implementation, interfaces shouldn't
- D. interfaces do not support abstraction, whereas an abstract class does

10. An abstract subclass can provide implementation of inherited abstract methods.

- True
- False

11. If A is a subclass of B, which of the following are true? Select all that apply.

- class B can access private members of class A
- objects of type A can be stored in variables of type B
- objects of type B can be stored in variables of type A
- class A inherits all non-private members of class B

12. Assume we are designing a BankAccount class. Which of the following methods are very likely to be static?

- A. deposit()

- B. getBalance()
C. convertToUSD()
D. getExchangeRate()
13. Which of the following are true of the final modifier? Select all that apply.
- a variable declared final can't be reassigned after initialization
 - a class declared as final cannot be subclassed
 - a final method cannot be overridden by subclasses
 - declaring a method as final will also make the class final
 - a final class must have a final constructor
 - a class marked as final must be abstract
-
-
-

14.

15.

16. Which of the following are true of abstract classes? Select all that apply.

- abstract classes are meant to be a framework for child classes
- abstract classes cannot be instantiated
- abstract classes cannot be marked final
- abstract classes cannot have a constructor

Week 3: Polymorphism and ADTs

17. What is polymorphism in Java?
- A. the property of an object to inherit more than one superclass at once
 - B. the ability of a method to modify its arguments directly
 - C. the act of defining multiple variables of the same type
 - D. the property of an object to take on multiple different forms
18. Which of the following are advantages of using polymorphism? Select all that apply.
- simplifies code maintenance by promoting modularity
 - enables a single interface to represent multiple behaviors
 - improves code scalability and flexibility

- guarantees faster execution speed
 - increased memory efficiency
 - code is easier to understand
19. Is this an example of method overloading, or method overriding?

```
class Food {  
    void prepare() {  
        System.out.println("Preparing food");  
    }  
}  
  
class Pasta extends Food {  
    void prepare() {  
        System.out.println("Mamma mia, pasta");  
    }  
}  
  
public static void main(String[] args) {  
    Food myFood = new Pasta();  
    myFood.prepare();  
}
```

- A. method overloading
- B. method overriding
- C. both
- D. neither

20. Is this an example of method overloading, or method overriding?

```
class Food {  
    void prepare() {  
        System.out.println("Preparing food...");  
    }  
  
    void prepare(String name) {  
        System.out.println("Preparing " + name + "...");  
    }  
}  
  
class Pizza extends Food {  
    void prepare() {  
        System.out.println("Preparing pizza...");  
    }  
  
    void prepare(String name, int size) {  
        System.out.println("Preparing " + size + "-in " + name);  
    }  
}
```

- A. method overloading
 - B. method overriding
 - C. both
 - D. neither
21. What is an abstract data type (ADT)?
- A. a data type defined by its implementation details
 - B. a data type defined by its behavior and operations, not its implementation
 - C. a specific type of data structure optimized for memory efficiency
 - D. a specific format for exporting and importing data between systems
22. What is the key characteristic of a Queue ADT?
- A. Last In, First Out (LIFO)
 - B. First In, First Out (FIFO)
 - C. Random Access
 - D. First Out, In Last (FOIL)
23. What distinguishes a LinkedList from an array?
- A. a linked list uses dynamic memory allocation
 - B. a linked list stores elements in a contiguous memory block
 - C. a linked list allows direct access to any element
 - D. a linked list has a fixed size
24. What is the primary advantage of using a LinkedList over an array?
- A. faster random access to elements
 - B. efficient insertion and deletion operations
 - C. ability to store only unique elements
 - D. it uses less memory than arrays
-
-
-

25. Week 5: GUIs, JavaFX, and MVC

26. Which of the following is NOT a feature of JavaFX?
- A. Styling using CSS
 - B. 2D and 3D graphics support
 - C. Integration with modern Java IDEs
 - D. Built-in database management system

27. What is the primary purpose of the Stage class in JavaFX?
- A. to define the layout of UI elements
 - B. to act as the main container for the JavaFX application window
 - C. to represent the controller of the application
 - D. to define the application's event listeners
28. What is the purpose of an event handler in JavaFX?
- A. To create new UI components
 - B. To manage user interactions like clicks and key presses
 - C. To manage data binding between the model and the view
 - D. To apply styling to JavaFX nodes
-
-
-

29.

30. In a JavaFX application using MVC, what would typically trigger a change in the Model?
- A. direct interaction with the Model by the View
 - B. a user action in the View, processed by the Controller
 - C. automatic updates from the View to the Model
 - D. the initialization of the Stage
31. What type of event does the setOnAction() method handle?
- A. Mouse events only
 - B. Keyboard events only
 - C. Action events like button clicks
 - D. Focus change events

Wrap-Up: Code Tracing

32. [Lecture 01, 02, 04] What will be the output of this code?

```
class Book {  
    String title;  
    double price;  
  
    Book(String title, double price) {  
        this.title = title;  
        this.price = price;  
    }  
}
```

```
public class Test {  
    public static void main(String[] args) {  
        Book book1 = new Book("how to be a sigma", 29.99);  
        Book book2 = new Book("how to be a sigma", 29.99);  
        System.out.print(book1 == book2);  
        System.out.print(book1.title.equals(book2.title));  
    }  
}
```

- A. falsefalse
- B. falsetrue
- C. truefalse
- D. truetrue

33. [Lecture 04, 07] What will be the output of this code?

```
class Course {  
    void enroll(Course course) {  
        System.out.println("Student enrolls in course.");  
    }  
    void enroll(CompSci course) {  
        System.out.println("Student enrolls in CS course.");  
    }  
}  
class CompSci extends Course {  
    void enroll(Course course) {  
        System.out.println("CS student enrolls in course.");  
    }  
    void enroll(CompSci course) {  
        System.out.println("CS student enrolls in CS course.");  
    }  
}  
public class Test {  
    public static void main(String[] args) {  
        Course c1 = new CompSci();  
        Course c2 = new Course();  
        c1.enroll(c2);  
        c2.enroll(c1);  
    }  
}
```

- A. CS student enrolls in course.
Student enrolls in course.
- B. Student enrolls in course.
CS student enrolls in course.
- C. Student enrolls in CS course.
Student enrolls in CS course.

- D. CS student enrolls in course.
Student enrolls in CS course.

34. [Lecture 06, 07] What will be the output of this code?

```
interface Club {  
    void join();  
}  
  
class SportsClub implements Club {  
    public void join() {  
        System.out.println("Joining a sports club.");  
    }  
}  
  
class MusicClub implements Club {  
    public void join() {  
        System.out.println("Joining a music club.");  
    }  
}  
  
public class Test {  
    public static void main(String[] args) {  
        Club club = new SportsClub();  
        MusicClub musicClub = (MusicClub) club;  
        musicClub.join();  
    }  
}
```

- A. Joining a sports club.
Joining a music club.
B. Joining a music club.
C. Joining a sports club.
D. Exception in thread "main"

35. [Lecture 08] What is the output of the following code?

```
import java.util.ArrayList;  
  
public static void main(String[] args) {  
    ArrayList<Integer> list = new ArrayList<>();  
    list.add(1);  
    list.add(2);  
    list.remove(1);  
    list.add(4);  
    list.add(1, 2);  
    System.out.println(list);  
}
```

- A. [1, 2, 4]
- B. [2, 2, 4]
- C. [1, 4, 1]
- D. [2, 4, 1]

36. [Lecture 09] What is the output of the following code?

```
import java.util.HashMap;

public static void main(String[] args) {
    HashMap<String, String> map = new HashMap<>();
    map.put("K", "Khushpreet");
    map.put("J", "Jack");
    map.put("K", "Kareem");
    System.out.println(map.get("K") + " " + map.size());
}
```

- A. Khushpreet 3
- B. Kareem 2
- C. Khushpreet 2
- D. Compilation Error

37. [Lecture 10] What is the output of the following code?

```
public class Main {
    public static void main(String[] args) {
        try {
            String str = "who is passing the exam";
            str = null;
            System.out.print(str.length());
        } catch (ArithmeticException e) {
            System.out.print("Ishaan, ");
        } catch (NullPointerException e) {
            System.out.print("Aaryan, ");
        } catch (Exception e) {
            System.out.print("Anushka, ");
        } finally {
            System.out.print("Ahmad, ");
        }
    }
}
```

- A. Ishaan, Aaryan, Anushka, Ahmad,
- B. Aaryan, Ahmad,
- C. Aaryan, Anushka, Ahmad
- D. Ahmad,

38. [Lecture 11] What is the output of the following code?

```
import java.io.*;

public static void main(String[] args) throws IOException {
    File IO Exception = new File("classlist.txt");
    file.createNewFile();

    try (PrintWriter writer = new PrintWriter(new
        FileWriter(file))) {
        writer.write("Chi M\n");
        writer.write("Bliss I");
    }

    try (BufferedReader reader = new BufferedReader(new
        FileReader(file))) {
        System.out.println(reader.readLine());
        System.out.println(reader.readLine());
        System.out.print(reader.readLine());
    }
}
```

- A. Chi M Bliss I null
- B. Chi M
Bliss I null
- C. Chi M
Bliss I
IOException
- D. IOException

39. [Lecture 04, 07, 09] What will be the output of this program?

```
import java.util.*;

class ClassMark {}
class Quiz extends ClassMark {}
class Tutorial extends ClassMark {}

public class Main {
    public static void main(String[] args) {
        Set<ClassMark> marks = new HashSet<>();
        marks.add(new Quiz());
        marks.add(new Tutorial());
        marks.add(new Quiz());

        System.out.println(marks.size());
    }
}
```

- A. 1
B. 2
C. 3
D. Compilation Error
40. [Lecture 04, 07] Which of the following lines cause an exception (i.e. are invalid) if left in? Select any that apply.

```
class Building {}  
class Residence extends Building {}  
class Stormont extends Residence {}  
class Herzberg extends Building {}  
class Southam extends Building {}  
  
public static void main(String[] args) {  
    Residence res = new Residence();  
    Building b1 = (Building) res;  
    Stormont storm = new Stormont();  
    Residence res2 = (Residence) storm;  
    Southam southam = (Southam) res2;  
    Herzberg herz = (Herzberg) b1;  
}
```

- line 9
- line 10
- line 11
- line 12
- line 13
- line 14

41. [Lecture 12, 13, 14] What should we replace line 19 with to achieve the intended functionality of this JavaFX program?

```
public class Main extends Application {  
    public static void main(String[] args) { launch(args); }  
  
    private int counter = 0;  
    public void start(Stage primaryStage) {  
        Label label = new Label("Count: 0");  
        Button button = new Button("Increase Count");  
  
        button.setOnAction(e -> {  
            counter++;  
  
            // i'm on break, fix it yourself  
       });
```

```
VBox layout = new VBox(10);
layout.getChildren().addAll(label, button);

primaryStage.setScene(new Scene(layout, 200, 150));
primaryStage.setTitle("Counter App");
primaryStage.show();
}
```

- A. counter = label.getText()
- B. label = new Label("Count: " + counter)
- C. label.text = "Count" + counter
- D. label.setText("Count: " + counter)