1. What will be the output of the following code?

```
class Test {
    void display(int a) {
        System.out.println("Primitive int: " + a);
    }
    void display(Integer a) {
        System.out.println("Wrapper Integer: " + a);
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.display(10);
    }
}
```

A) Primitive int: 10
B) Wrapper Integer: 10
C) Compilation error
D) Runtime error

---

2. Which method will be called when passing `null`?

```
class Test {
    void display(Integer a) {
        System.out.println("Wrapper Integer");
    }
    void display(Double a) {
        System.out.println("Wrapper Double");
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.display(null);
    }
}
```

A) Wrapper Integer
B) Wrapper Double
C) Compilation error
D) Runtime error

---

3. What happens if we overload a method with `int` and `long`, and pass `10`?

A) The `int` version is called
B) The `long` version is called
C) Compilation error
D) Runtime error

---

4. Can we overload a method by changing only the return type?

A) Yes
B) No

C) Only if the methods are `static`
D) Only inside an abstract class

---

5. What will be the output of the following code?

```java
class Test {
    void show(Number n) {
        System.out.println("Number method");
    }
    void show(Integer i) {
        System.out.println("Integer method");
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.show(10);
    }
}
```

A) Number method
B) Integer method
C) Compilation error
D) Runtime error

---

6. What will be the output of the following program?

```java
class Overload {
    void print(Double d) {
        System.out.println("Double method");
    }
    void print(Float f) {
        System.out.println("Float method");
    }
    public static void main(String[] args) {
        Overload obj = new Overload();
        obj.print(10.5);
    }
}
```

A) Double method
B) Float method
C) Compilation error
D) Runtime error

---

7. Which method will be called when passing `10L`?

```java
void show(int a)
void show(long a)
```

A) `show(int a)`
B) `show(long a)`

C) Compilation error
D) Runtime error

---

8. What will be the output of the following code?

```
class Overload {
    void display(Double d) {
        System.out.println("Double method");
    }
    void display(Integer i) {
        System.out.println("Integer method");
    }
    public static void main(String[] args) {
        Overload obj = new Overload();
        obj.display(10);
    }
}
```

A) Double method
B) Integer method
C) Compilation error
D) Runtime error

---

9. Which of the following overloaded methods will be chosen if we pass `null`?

```
void show(String s)
void show(Object o)
```

A) `show(String s)`
B) `show(Object o)`
C) Compilation error
D) Runtime error

---

10. What happens if we overload a method with `int` and `Integer`, and we pass `null`?

A) Calls the `int` method
B) Calls the `Integer` method
C) Compilation error
D) Runtime error

---

11. What will be the output of the following code?

```
class Overload {
    void show(int a) {
        System.out.println("int method");
    }
    void show(Integer a) {
        System.out.println("Integer method");
    }
    public static void main(String[] args) {
        Overload obj = new Overload();
        Integer num = 10;
        obj.show(num);
    }
}
```

A) int method
B) Integer method
C) Compilation error
D) Runtime error

---

12. Can overloaded methods have different return types if their parameters differ?

A) Yes
B) No
C) Only in abstract classes
D) Only for `static` methods

---

13. What will be the output of the following code?

```
class Test {
    void show(Integer i) {
        System.out.println("Integer method");
    }
    void show(Double d) {
        System.out.println("Double method");
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.show(10.0);
    }
}
```

A) Integer method
B) Double method
C) Compilation error
D) Runtime error

---

14. Can overloaded methods be defined in different classes?

A) Yes, if they are related through inheritance
B) No, they must be in the same class
C) Only if they are `static`
D) Only if they are `private`

---

15. What will be the output of the following program?

```
class A {
    void display(Number n) {
        System.out.println("Number method");
    }
}
class B extends A {
    void display(Double d) {
        System.out.println("Double method");
    }
    public static void main(String[] args) {
        B obj = new B();
        obj.display(10);
    }
}
```

A) Number method
B) Double method
C) Compilation error
D) Runtime error

---

16. What will be the output of the following program?

```
class A {
    void show(Object o) {
        System.out.println("Object method");
    }
}
class B extends A {
    void show(String s) {
        System.out.println("String method");
    }
    public static void main(String[] args) {
        A obj = new B();
        obj.show(null);
    }
}
```

A) Object method
B) String method
C) Compilation error
D) Runtime error

---

17. What happens if we overload methods with `Integer`, `Long`, and `Double`, and pass `10`?

A) Calls `Integer` method
B) Calls `Long` method
C) Calls `Double` method
D) Compilation error

---

18. What will be the output of the following program?

```java
class Test {
    void display(int a, long b) {
        System.out.println("int-long method");
    }
    void display(long a, int b) {
        System.out.println("long-int method");
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.display(10, 10L);
    }
}
```

A) int-long method
B) long-int method
C) Compilation error
D) Runtime error

---

19. Can `final` methods be overloaded?

A) Yes
B) No
C) Only if they are `static`
D) Only if they return `void`

---

20. What will be the output of the following code?

```java
class Test {
    void show(int a) {
        System.out.println("int method");
    }
    void show(Number n) {
        System.out.println("Number method");
    }
    public static void main(String[] args) {
        Test obj = new Test();
        obj.show(10);
    }
}
```

A) int method
B) Number method
C) Compilation error
D) Runtime error

Scenario question

1. Payment Processing System

- **Method Name:** `makePayment`
- **Parameters:**

    - `makePayment(String cardNumber, int cvv)` → Payment using a card
    - `makePayment(String upiId)` → Payment using UPI
    - `makePayment(String accountNumber, String ifsc)` → Payment using bank account

Explanation:

A payment system allows users to make payments through different methods: **credit/debit cards, UPI, or bank transfer**. The method is overloaded to handle different payment types based on input parameters.

---

2. Employee Salary Calculation

- **Method Name:** `calculateSalary`
- **Parameters:**
    - `calculateSalary(int hoursWorked, double hourlyRate)` → Hourly employee salary
    - `calculateSalary(double monthlySalary)` → Fixed salary employee
    - `calculateSalary(int projectsCompleted, double paymentPerProject)` → Freelancer salary

Explanation:

Different employees have different salary structures (hourly, fixed, freelance). Overloading ensures each type of salary can be computed based on input parameters.

---

3. E-commerce Discount System

- **Method Name:** `applyDiscount`
- **Parameters:**
    - `applyDiscount(double price, double flatDiscount)` → Flat discount (e.g., $10 off)
    - `applyDiscount(double price, int percentage)` → Percentage discount (e.g., 10% off)
    - `applyDiscount(double price, int percentage, boolean isMember)` → Extra discount for premium members

Explanation:

An online shopping platform offers discounts in different ways (flat, percentage-based, special member). Overloading allows flexibility based on the type of discount.
