by Kunal Sir

Assignment : Polymorphism (Narrowing Concept)

Part A : MCQs

MCQ 1

What is the narrowing concept in polymorphism mainly related to?
A. Method overriding
B. Method overloading resolution
C. Type casting
D. Interfaces

Correct Answer: B

MCQ 2

Why does Java call the child-type method in narrowing?
A. Child class has more methods
B. Child type is more specific
C. Parent methods are ignored
D. Due to inheritance only

Correct Answer: B

MCQ 3

What will be the output?

```
class A {
    void m1(Object o) { System.out.println("Object"); }
    void m1(String s) { System.out.println("String"); }
}
public class Test {
    public static void main(String[] args) {
        new A().m1(null);
    }
}
```

A. Object
B. String
C. Compilation error
D. Runtime error

Correct Answer: B

MCQ 4

Which condition is mandatory for narrowing to occur?
A. Same return type
B. Same method body
C. Parent-child relationship between parameters
D. Use of casting

Correct Answer: C

MCQ 5

What happens if two overloaded methods have unrelated parameter types and `null` is passed?
A. Child method is called
B. Parent method is called
C. Compilation error
D. Runtime exception

Correct Answer: C

MCQ 6

Which of the following best describes narrowing?
A. JVM behavior at runtime
B. Compiler-time decision
C. Memory optimization technique
D. Exception handling mechanism

Correct Answer: B

MCQ 7

Which of the following arguments most commonly demonstrates narrowing?
A. int value
B. String literal
C. null
D. boolean value

Correct Answer: C

MCQ 8

If method parameters are `A` and `Object`, which one will be chosen?
A. Object
B. A
C. Randomly chosen
D. Causes error

Correct Answer: B

MCQ 9

Does narrowing depend on reference type or object type?
A. Object type
B. Reference type
C. Both
D. None

Correct Answer: B

MCQ 10

Which concept is MOST confused with narrowing?
A. Abstraction
B. Downcasting
C. Encapsulation
D. Multithreading

Correct Answer: B

Part B : Problem Statements

Problem 1: Notification System

```java
class Notification {
    void send(Object msg) {
        System.out.println("Generic Notification");
    }
    void send(String msg) {
        System.out.println("Text Notification");
    }
}
public class Test {
    public static void main(String[] args) {
        Notification n = new Notification();
        n.send(null);
    }
}
```

Task: Write the output and justify the answer using narrowing concept.

Output:

Text Notification

Reason: Both `Object` and `String` methods are applicable for `null`. `String` is more specific, so narrowing selects it.

Problem 2: Payment Processing

```
class Payment {
    void process(Object o) {
        System.out.println("Processing Payment");
    }
    void process(String mode) {
        System.out.println("Processing Payment Mode");
    }
}
public class Test {
    public static void main(String[] args) {
        new Payment().process(null);
    }
}
```

Task: Write the output and explain why that method is selected.

Output:

Processing Payment Mode

Reason: `String` is the child class of `Object`. Compiler applies narrowing and calls the `String` version.

Problem 3: Customer Support System

```
class Support {
    void help(Object o) {
        System.out.println("General Support");
    }
    void help(String issue) {
        System.out.println("Issue-Based Support");
    }
}
public class Test {
    public static void main(String[] args) {
        Support s = new Support();
        s.help(null);
    }
}
```

Task: Write the output and identify the narrowed parameter.

Output:

Issue-Based Support

Reason: `null` matches both methods, but `String` is more specific than `Object`.

Problem 4: Banking Application

```
class Bank {
    void openAccount(Object o) {
        System.out.println("Opening Generic Account");
    }
    void openAccount(String type) {
        System.out.println("Opening Savings Account");
    }
}
public class Test {
    public static void main(String[] args) {
        new Bank().openAccount(null);
    }
}
```

Task: Predict the output and explain the compiler decision.

Output:

Opening Savings Account

Reason: Narrowing rule prefers `String` parameter over `Object`.

Problem 5: E-Commerce Order System

```
class Order {
    void place(Object o) {
        System.out.println("Placing Order");
    }
    void place(String product) {
        System.out.println("Placing Product Order");
    }
}
public class Test {
    public static void main(String[] args) {
        Order o = new Order();
        o.place(null);
    }
}
```

Task: Write the output and explain the narrowing rule applied.

Output:

Placing Product Order

Reason: Among overloaded methods, `String` is the narrower type.

Problem 6: Hospital Management

```
class Hospital {
    void admit(Object o) {
        System.out.println("General Admission");
    }
    void admit(String patient) {
        System.out.println("Patient Admission");
    }
}
public class Test {
    public static void main(String[] args) {
        new Hospital().admit(null);
    }
}
```

Task: Write the output and explain why no ambiguity occurs.

Output:

Patient Admission

Reason: Compiler selects the child-type parameter (`String`) during method overloading.

Problem 7: Transport Booking

```
class Transport {
    void book(Object o) {
        System.out.println("Booking Transport");
    }
    void book(String vehicle) {
        System.out.println("Booking Vehicle");
    }
}
public class Test {
    public static void main(String[] args) {
        Transport t = new Transport();
        t.book(null);
    }
}
```

Task: Identify the output and justify using narrowing.

Output:

Booking Vehicle

Reason: `String` parameter is more specific than `Object`, so narrowing occurs.

Problem 8: Online Learning Platform

```
class Course {
    void enroll(Object o) {
        System.out.println("General Enrollment");
    }
    void enroll(String courseName) {
        System.out.println("Course Enrollment");
    }
}
public class Test {
    public static void main(String[] args) {
        new Course().enroll(null);
    }
}
```

Task: Write the output and explain the specificity rule.

Output:

Course Enrollment

Reason: Narrowing chooses the most specific applicable method (`String`).

Problem 9: Social Media App

```
class Post {
    void share(Object o) {
        System.out.println("Sharing Content");
    }
    void share(String text) {
        System.out.println("Sharing Text Post");
    }
}
public class Test {
    public static void main(String[] args) {
        Post p = new Post();
        p.share(null);
    }
}
```

Task: Write the output and explain method selection.

Output:

Sharing Text Post

Reason: `String` is a subclass of `Object`, hence selected by compiler.

Problem 10: Travel Booking System

```
class Travel {
    void book(Object o) {
        System.out.println("Booking Travel");
    }
    void book(String destination) {
        System.out.println("Booking Destination");
    }
}
public class Test {
    public static void main(String[] args) {
        new Travel().book(null);
    }
}
```

Task: Write the output and justify using narrowing concept.

Output:

Booking Destination

Reason: The compiler performs narrowing and calls the `String` version.
