Flynaut SaaS Pvt. Ltd.

Phase I

Puzzle-Based MCQs

"One always wants to be noticed by the compiler, the other doesn't care. Who are they?"

Options:

A. Error & Exception

B. IOException & NullPointerException

C. ClassNotFoundException & AssertionError

D. RuntimeException & SQLException

What will happen if you write a try block without a catch or finally?

Options:

A. Code compiles and runs fine
B. Compilation error
C. Runtime error
D. Infinite loop

In a Java court, who always gets the final word?

Options:

A. Try

B. Catch

C. Finally

D. Throw

Which keyword is used to manually trigger an exception?

Options:

A. throws

B. new

C. throw

D. raise

I look harmless but crash your program at runtime. I’m not checked at compile time, yet I love to strike. Who am I?

Options:

A. IOException
B. NullPointerException
C. SQLException
D. FileNotFoundException

I look like a class, but you can’t create my object. I guide other classes on how to behave. Who am I?

Options:

A. Interface
B. Abstract Class
C. Enum
D. Static Class

I run once even if the condition is false. Who am I?

Options:

A. for loop
B. while loop
C. do-while loop
D. enhanced for loop

I accept only unique elements and give no guarantee on order.

Options:

A. ArrayList
B. LinkedList
C. HashSet
D. TreeMap

I define ‘what’ not ‘how’. I am a contract, not a class.

Options:

A. Interface
B. Abstract Class
C. Class
D. Final Class

I maintain insertion order. Who am I?

Options:

A. HashMap
B. TreeMap
C. LinkedHashMap
D. Hashtable

You cannot touch my variables directly, but I give you getters and setters. Who am I?

Options:

A. Inheritance
B. Polymorphism
C. Abstraction
D. Encapsulation

All classes in Java secretly inherit me. Who am I?

Options:

A. Object
B. Class
C. Interface
D. Static

I have the same name but different parameters. Who am I?

Options:

A. Method Overriding
B. Method Hiding*
C. Method Overloading
D. Abstract Method

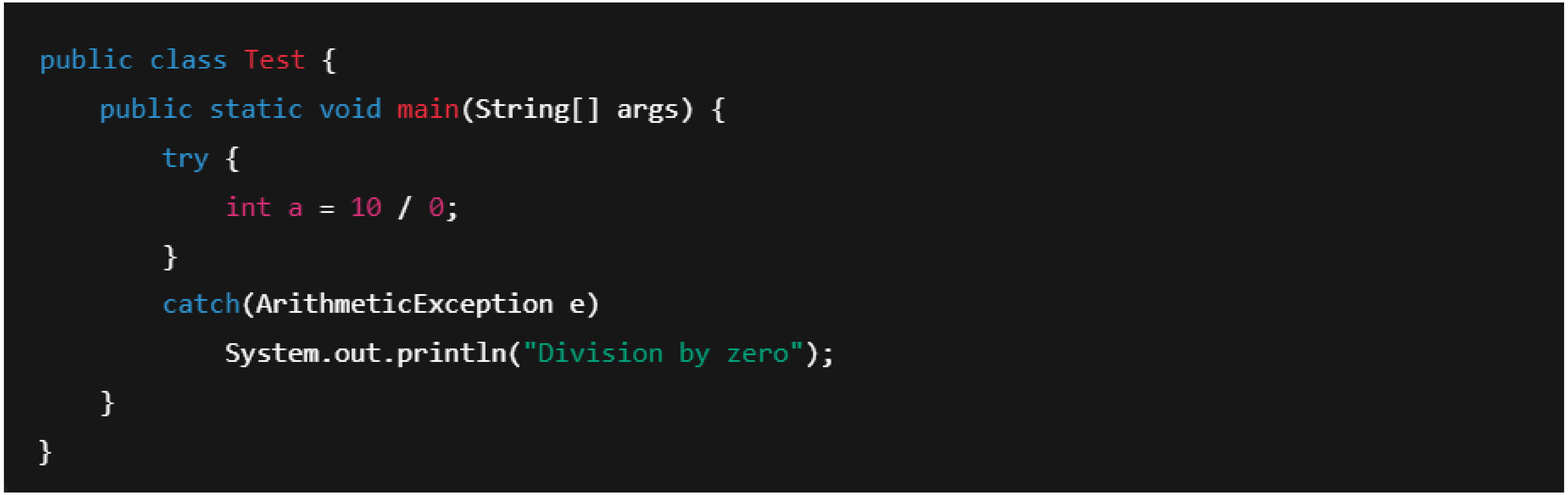
I'm inherited but redefined. When you call me on a parent reference, I behave like the child. Who am I?

Options:

A. Overloaded Method
B. Final Method
C. Overridden Method
D. Static Method

Phase II

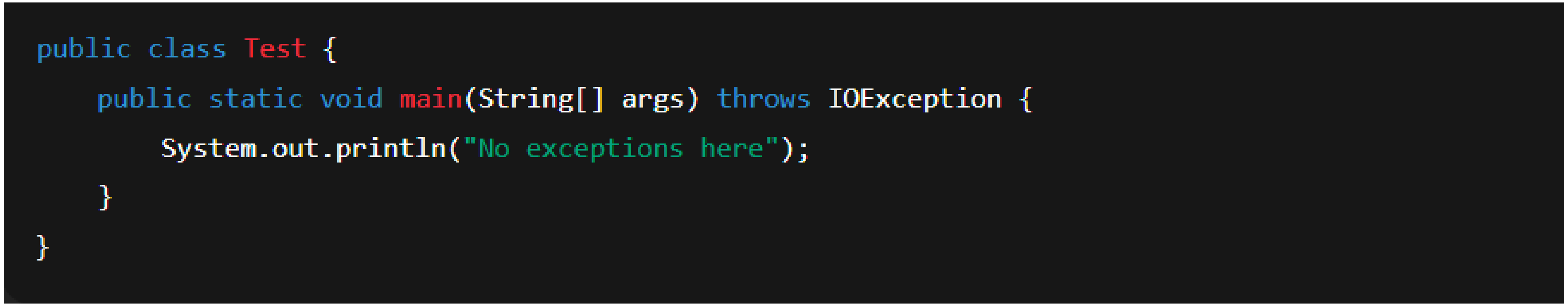
Guess the Error



```
public class Test {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        }
        catch(ArithmeticException e)
            System.out.println("Division by zero");
    }
}
```

Options:

A. No error
B. Missing throw statement
C. [✓ Missing braces in catch block]
D. Cannot divide int by 0



```java
public class Test {
    public static void main(String[] args) throws IOException {
        System.out.println("No exceptions here");
    }
}
```

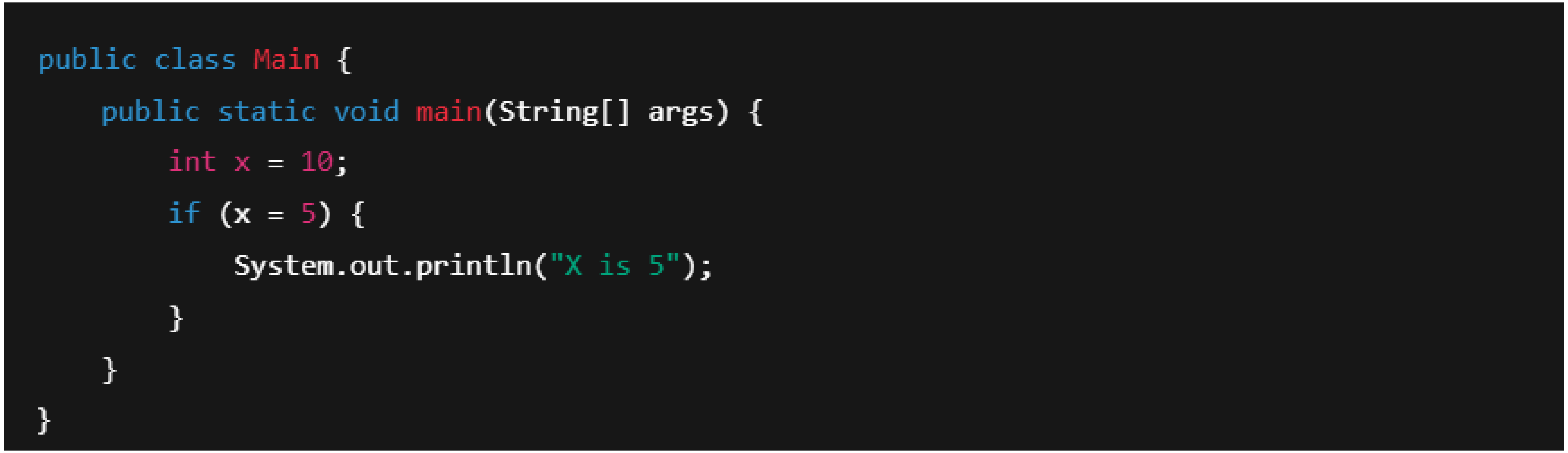
Options:

A. Compile error: IOException not handled

B. Runtime error

C. [✓ No error]

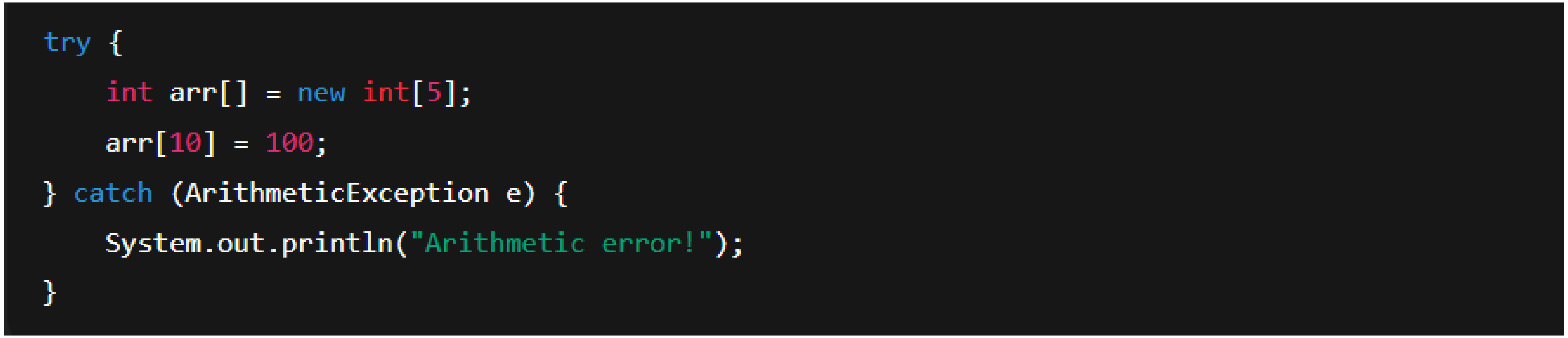
D. IOException is unchecked



```
public class Main {
    public static void main(String[] args) {
        int x = 10;
        if (x = 5) {
            System.out.println("X is 5");
        }
    }
}
```

Options:

A. No error
B. x can’t be used in if condition
C. = used instead of ==
D. Variable x not initialized



```java
try {
    int arr[] = new int[5];
    arr[10] = 100;
} catch (ArithmeticException e) {
    System.out.println("Arithmetic error!");
}
```

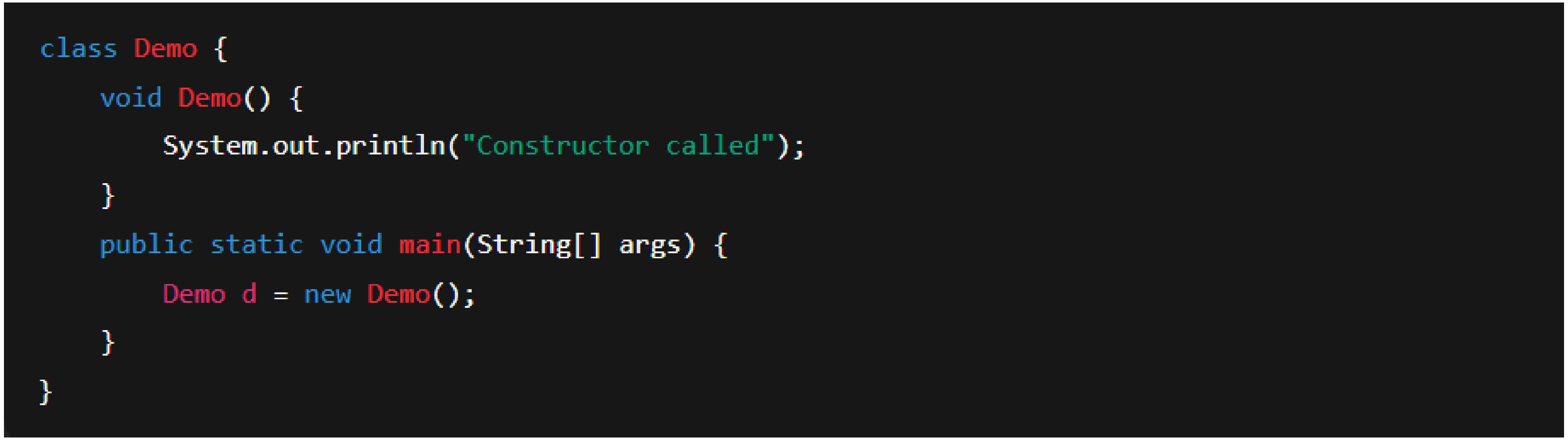
Options:

A. Compile error

[✓ B. ArrayIndexOutOfBoundsException not caught.]

C. ArithmeticException is thrown

D. Program runs fine



```
class Demo {
    void Demo() {
        System.out.println("Constructor called");
    }
    public static void main(String[] args) {
        Demo d = new Demo();
    }
}
```

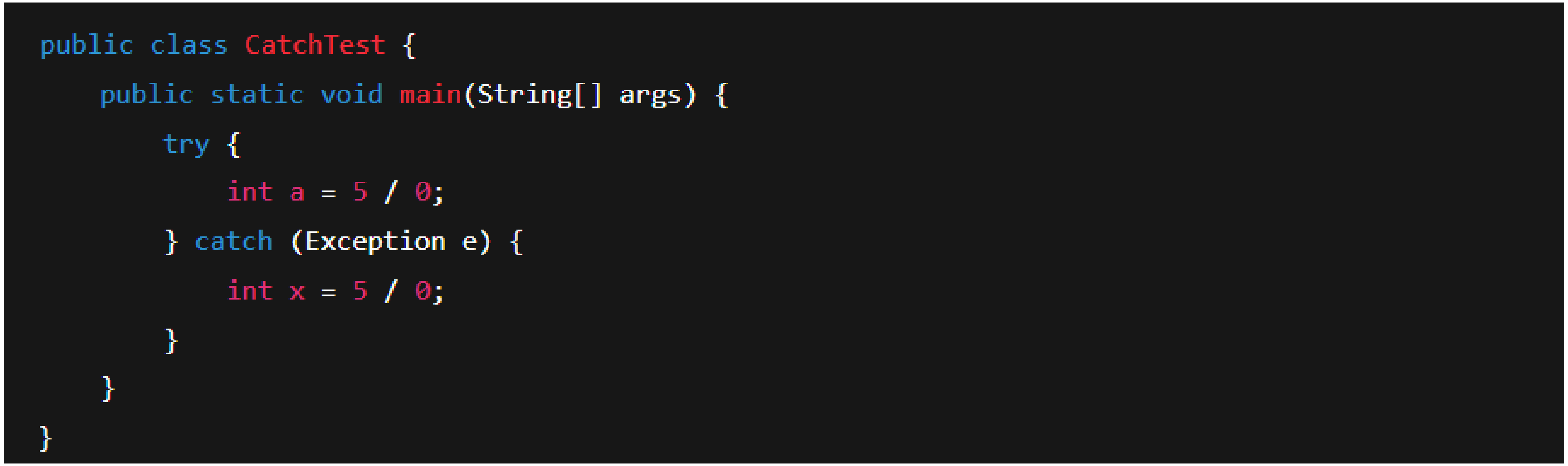
Options:

A. Constructor called

B. Compilation error

[✓ C. No output]

D. Runtime exception



```
public class CatchTest {
    public static void main(String[] args) {
        try {
            int a = 5 / 0;
        } catch (Exception e) {
            int x = 5 / 0;
        }
    }
}
```

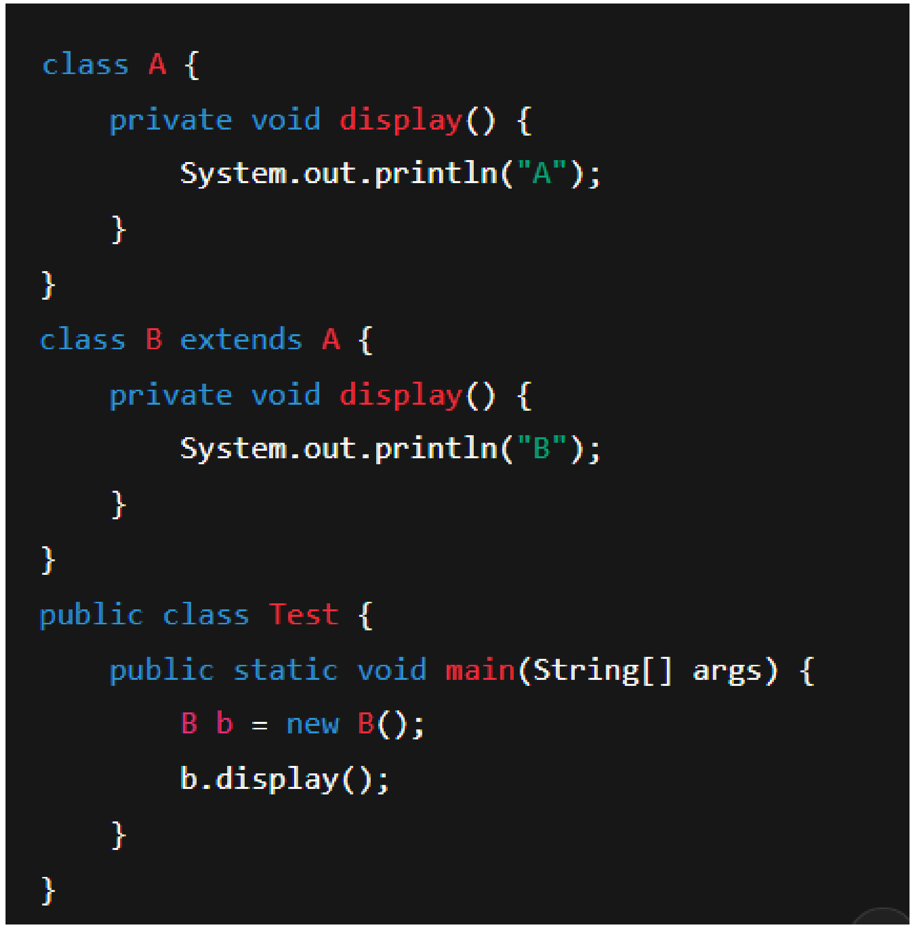
Options:

A. Compile error

B. Catch block handles all

C. Unhandled exception in catch

D. Program runs fine



```
class A {
    private void display() {
        System.out.println("A");
    }
}
class B extends A {
    private void display() {
        System.out.println("B");
    }
}
public class Test {
    public static void main(String[] args) {
        B b = new B();
        b.display();
    }
}
```

Options:
A. A
B. B
C. Compile error
D. Method hiding, not overriding

Phase III

Implementation Questions

Problem Statement:

Ask the user for age.
If age < 16, throw a custom exception DrivingNotAllowedException.

Else, print “You can apply for a driving license”

Problem Statement:

Ask for the nationality of the user.
If not “Indian”, throw a custom exception InvalidCitizenException

If valid, print “Eligible to vote”

Problem Statement:

Accept marks as input.

If marks are not in range 0 to 100, throw a custom exception InvalidMarksException.

Otherwise, print ″Marks recorded″.

Problem Statement:

Write a program to validate password strength.

If the password length is less than 8 characters, throw a custom exception WeakPasswordException.

If valid, print "Password accepted".