Java MCQs - Constructors and Exception Handling

Q1. Which of the following is true about Java constructors?

A. Constructors have a return type of void.

B. Constructors must always be public.

C. A constructor is automatically called when an object is created.

D. Constructors can only be defined explicitly.

Q2. What happens if you do not define a constructor in a Java class?

A. The program will not compile.

B. The compiler provides a default constructor.

C. The class cannot be instantiated.

D. The constructor from another class will be used.

Q3. What will be the output of the following code?

```
class Test {
    int x;
    Test() {
        x = 10;
    }
}
public class Main {
    public static void main(String[] args) {
        Test obj = new Test();
        System.out.println(obj.x);
    }
}
```

A. Compilation error

B. 0

C. 10

D. Null

Q4. Can a Java constructor be private?

A. No, it must be public.

B. Yes, but the class cannot be instantiated outside the class.

C. Yes, and it can be accessed from anywhere.

D. No, Java does not allow private constructors.

Q5. How can one constructor call another constructor within the same class?

A. Using super()

B. Using this()

C. Using new()

D. Constructors cannot call each other

Q6. What will be the output of this code?

```
class Demo {
    int num;
    Demo() {
        this(100);
        System.out.println("Default Constructor");
    }
    Demo(int n) {
        num = n;
        System.out.println("Parameterized Constructor");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Demo obj = new Demo();
    }
}
```

A. Default Constructor

B. Parameterized Constructor

C. Parameterized Constructor Default Constructor

D. Compilation Error

Q7. What will happen if you explicitly define a constructor with parameters but do not define a no-argument constructor?

A. Java will provide a default no-argument constructor.

B. You cannot create an object without passing arguments.

C. Compilation error occurs.

D. The constructor will be inherited from Object class.

Q8. Can a constructor be final in Java?

A. Yes, if you override it.

B. No, because constructors are not inherited.

C. Yes, but only in abstract classes.

D. Yes, but only for utility classes.

Q9. What will be the output of the following code?

```
class Base {
    Base() {
        System.out.println("Base Constructor");
    }
```

```
}
class Derived extends Base {
    Derived() {
        System.out.println("Derived Constructor");
    }
}
public class Main {
    public static void main(String[] args) {
        Derived obj = new Derived();
    }
}
```

A. Base Constructor Derived Constructor

B. Derived Constructor Base Constructor

C. Compilation Error

D. Runtime Error

Q10. What will be the output of this program?

```
class A {
    private A() {
        System.out.println("Private Constructor");
    }
    public static void createInstance() {
        new A();
    }
}
public class Main {
```

```
    public static void main(String[] args) {
        A.createInstance();
    }
}
```

A. Compilation Error

B. Private Constructor

C. Runtime Error

D. No Output

Q11. Which statement about final, finally, and finalize is correct?

A. final is used to handle exceptions, finally is used for garbage collection, and finalize prevents modificatio

B. final makes a variable constant, finally ensures execution of cleanup code, and finalize is called before g

C. final is a method, finally is a keyword, and finalize is a block inside try-catch.

D. final, finally, and finalize all serve the same purpose.

Q12. Which of the following blocks always executes, regardless of an exception occurring or not?

A. try

B. catch

C. finally

D. throw

Q13. What will be the output of the following code?

```
public class Geeks {
    public static void main(String[] args) {
        try {
            System.out.println("Inside try");
            throw new RuntimeException("Error");
```

```
        } finally {
            System.out.println("Inside finally");
        }
    }
}
```

A. Inside try

B. Inside try Inside finally

C. Inside try Inside finally RuntimeException

D. Compilation Error

Q14. What will happen in the following code?

```
public class Geeks {
    static void method() throws Exception {
        throw new Exception("Error occurred");
    }

    public static void main(String[] args) {
        method();
    }
}
```

A. Compilation Error

B. Runtime Exception

C. Exception: Error occurred

D. Program runs successfully

Q15. What is the difference between throw and throws?

A. throw is used to declare exceptions, throws is used to throw exceptions

B. throws is used to declare exceptions, throw is used to throw exceptions

C. Both are used to declare exceptions

D. Both are used to throw exceptions

Q16. What will be the output of the following program?

```
class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}

public class Geeks {
    public static void main(String[] args) {
        try {
            throw new CustomException("Custom error occurred");
        } catch (CustomException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

A. Custom error occurred

B. Runtime Exception

C. Compilation Error

D. No Output

Q17. What happens when finalize() is called on an object?

A. The object is immediately garbage collected.

B. The garbage collector calls it before collecting the object.

C. The object gets permanently deleted from memory.

D. It prevents an object from being collected.

Q18. What is the output of the following code?

```
public class Geeks {
    public static void main(String[] args) {
        try {
            System.exit(0);
        } finally {
            System.out.println("Finally executed");
        }
    }
}
```

A. Finally executed

B. No output

C. Runtime Error

D. Compilation Error

Q19. Which of the following is true about custom exceptions?

A. Custom exceptions cannot extend Exception class

B. Custom exceptions are always checked exceptions

C. Custom exceptions can extend Exception or RuntimeException

D. Custom exceptions do not require a constructor

Q20. Which of the following correctly defines a custom exception?

A. class MyException { public MyException(String message) { super(message); } }

B. class MyException extends RuntimeException { public MyException(String message) { super(message);

C. class MyException extends Throwable { public MyException(String message) { super(message); } }

D. class MyException extends Error { public MyException(String message) { super(message); } }