2. Comments, Constructors, and Main method

Comments

1. Comments are textual annotations that provide additional information about the code without affecting its execution
2. // This is a single-line comment
3. /* * This is a multi-line comment

   It can span multiple lines */

4. Javadoc comments are special type of comments used to generate java documentation

/** * This is a Javadoc comment.

* It provides information about the purpose and usage of class and methods. */

QUIZ

Which of the following is not a valid code comment in Java?

A. // Add 5 to the result

B. /*** TODO: Fix bug 12312 ***/

C. # Add configuration value

D. /* Read file from system ****/

QUIZ ANSWER

Which of the following is not a valid code comment in Java?

A. // Add 5 to the result

B. /*** TODO: Fix bug 12312 ***/

C. # Add configuration value

D. /* Read file from system ****/

Java accepts Options A, B, and D as valid comments. Note that the /* */ syntax can have additional (and uneven) star (*) characters as shown in B and D. Option C is incorrect as hashtag (#) is not a valid comment character in Java.

Methods

1. A method is a block of code that performs a specific task or operation.
2. Methods are defined within a class and can be called to execute the code they contain.
3. They allow you to organise code into reusable units, making your program more modular and easier to maintain.

Methods

```
// In this example, sayHello is a method defined within the MyClass class
public class MyClass {
    public void sayHello() {
       //prints "Hello, World!" to the console
         System.out.println("Hello, World!”);
    }
   // The main method creates an instance of MyClass
    public static void main(String[] args) {
        MyClass myObject = new MyClass();
        myObject.sayHello(); // Call the sayHello method
    }
}
```

Methods

1. Methods can have return types (such as int, double, String, or void for no return value) and parameters (inputs) that allow you to pass data into the method.

Methods

```
public class Calculator {
        // add method takes two int parameters
    public int add(int a, int b) {
        return a + b; // return sum
    }
    public static void main(String[] args) {
                // create instance of Calculator
        Calculator calculator = new Calculator();
        int result = calculator.add(5, 3); // Call the add method
        System.out.println("Result: " + result);
    }
}
```

QUIZ

Structuring a Java class such that only methods within the class can access its instance variables is referred to as?

A. platform independence

B. object orientation

C. inheritance

D. encapsulation

QUIZ ANSWER

Structuring a Java class such that only methods within the class can access its instance variables is referred to as?

A. platform independence

B. object orientation

C. inheritance

D. encapsulation

Encapsulation is the technique of removing access to a class's instance variables from processes outside the class, making Option D the correct answer

Constructors

1. In Java, a constructor is a special type of method that is used to initialise objects.
2. It has the same name as the class and does not have a return type, not even void.
3. Constructors are called when an object of a class is created using the new keyword.

Constructors

```
        // takes two parameters:
        // name (String representing the name)
// age (int representing the age)

        public MyClass(String name, int age) {
            // this keyword is used to refer to the current instance
            // of the MyClass object being created
    this.name = name;
    this.age = age;
            // initialised with a new ArrayList object
    this.hobbies = new ArrayList<>();
}
```

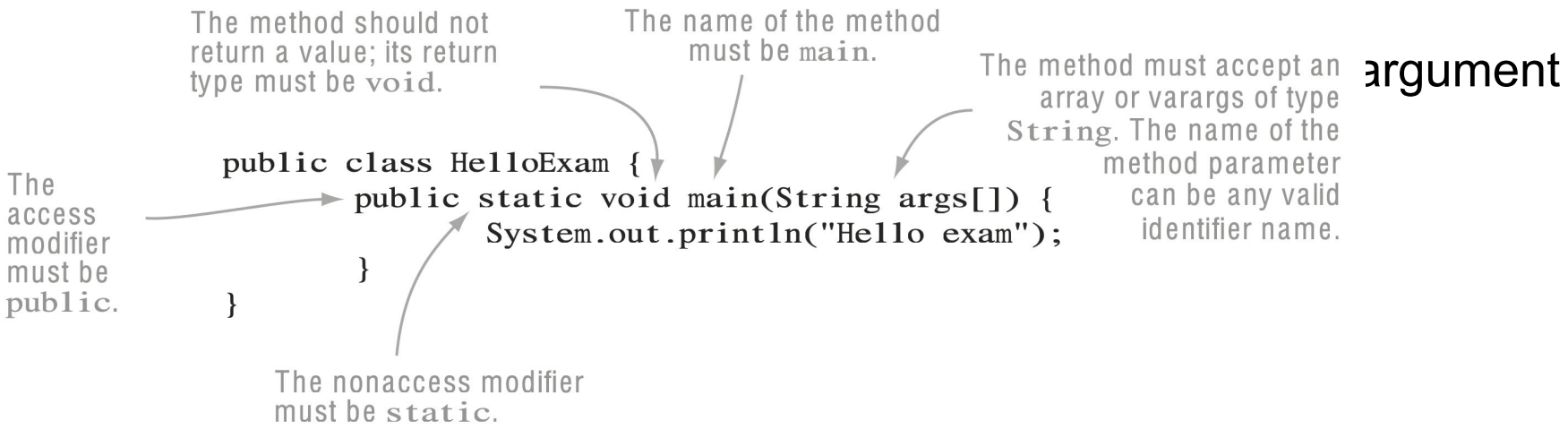
Executable Classes

1. An executable Java class, when handed over to the JVM, starts its execution at a particular point in the class—the main method.
2. The JVM starts executing the code that's defined in the main method.
3. You can't hand over a non-executable Java class (class without a main method) to the JVM and ask it to execute it.
4. In this case, the JVM won't know which method to execute because no entry point is marked.

Main Method

1. Main method should comply with the following rules:

   1. The method must be marked as a public method.
   2. The method must be marked as a static method.
   3. The name of the method must be main.
   4. The return type of this method must be void.
   5. The method (varargs) of argument



Main Method

1. It's valid to define the method parameter passed to the main method as a variable argument (varargs) of type String:

```
public static void main (String... args)
```

It's valid to define args as a variable argument.

Main Method

1. To define a variable argument variable, the ellipsis (...) must follow the type of the variable and not the variable itself (a mistake made by a lot of new programmers):

```
public static void main (String args…)
```

This won't compile. Ellipsis must

follow the data type, String.

Main Method

1. As mentioned previously, the name of the String array passed to the main method need not be args to qualify it as the correct main method.
2. The following examples are also correct definitions of the main method:

public static void main(String[] arguments)

public static void main(String[] Helloworld)

The names of the method arguments are

arguments and Hello World, which is acceptable.

Main Method

1. To define an array, the square brackets [ ] can follow either the variable name or its type. The following is a correct method declaration of the main method:
2. The following examples are also correct definitions of the main method:

public static void main(String[] args)

public static void main (String minnieMouse[])

The square brackets [] can follow either

the variable name or its type.

Main Method

1. It's interesting to note that the placement of the keywords public and static can be interchanged, which means that the following are both correct method declarations of the main method:

public static void main (String[] args)

static public void main(String[] args)

The placements of the keywords

public and static are interchangeable.

NOTE: Though both public static and static public are the valid order of keywords to declare the main method, public static is more common and thus more readable.

QUIZ

Which of the following method signatures is a valid declaration of an entry point in a Java application?

A. public void main(String[] args)

B. public static void main()

C. private static void start(String[] mydata)

D. public static final void main(String[] mydata)

QUIZ ANSWER

Which of the following method signatures is a valid declaration of an entry point in a Java application?

A. public void main(String[] args)

B. public static void main()

C. private static void start(String[] mydata)

D. public static final void main(String[] mydata)

An entry point in a Java application consists of a main method with a single String[] argument, return type of void, and modifiers public and static.

The name of the variable in the input argument does not matter.

Option A is missing the static modi-fier, Option B is missing the String[] argument, and Option C has the wrong access modifier and method name. Only Option D fulfills these requirements. Note that the modifier final is optional and may be added to an entry point method.

Java 21 Feature(Not required for exam)

1. In Java 21 you can write main method more simply.
2. This is a new feature in Java and not required for your exam
3. In java 21, you can start your program without having a class, and with less keywords than previously.
4. This feature called unnamed classes
5.
```
void main() {
System.out.println("Hello, World!");
}
```

Further reading(Optional):

https://www.baeldung.com/java-21-unnamed-class-instance-main

The use of String[] in main method

1. It allows you to pass data or parameters to your Java program from the command line. These parameters can be used to customize the behavior of the program without modifying its source code.

2. You can pass file location, URL, text etc

Run a Java program from the command line

1. Write Your Java Code
2. Open a terminal or command prompt window.
3. Navigate to the Directory
4. Use the `javac` command followed by the name of your Java file to compile it. For example, to compile `HelloWorld.java`, you would use:
    1. javac HelloWorld.java
    2. If there are no syntax errors in your Java file, this command will generate a `HelloWorld.class` file in the same directory.
5. After successfully compiling the Java file, you can run the Java program using the `java` command followed by the name of the class (without the `.class` extension). For example, to run the `HelloWorld` class, you would use:
    1. java HelloWorld

QUIZ

Which statements about Java are true?

1- The java command can execute .java and .class files.

2- Java is not object oriented.

3- The javac command compiles directly into native machine code.

A. 1 only

B. 2 only

C. 2 and 3

D. None are true.

QUIZ ANSWER

Which statements about Java are true?

The java command can execute .java and .class files.

Java is not object oriented.

The javac command compiles directly into native machine code.

A. I only
B. III only
C. II and III
D. None are true.

The java command can only execute compiled .class files, so I is false. Java is most certainly object oriented, one of the key design principles, so II is also false. The javac command compiles into bytecode, which must be run in a Java virtual machine (JVM), and is not native machine code, so III is false as well. Since none of the statements are true,