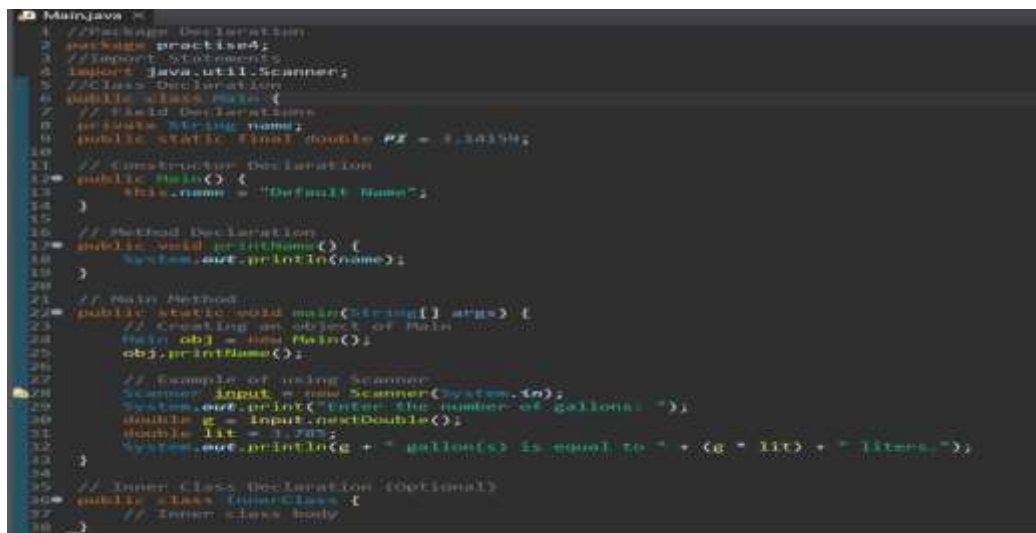


1. Name the components that comprise a .java file. List the components in the order that you would expect to see them in a Java program.

A. A java file typically contains several components that are structured in a specific order to form a valid Java program. Here are the components listed in the order that they are usually expected to appear:

1. Package Declaration
2. Import Statements
3. Class or Interface Declaration
4. Field Declarations (Instance and Static Variables)
5. Constructor Declarations
6. Inner Class or Interface Declarations



```
1 //Package Declaration
2 package practice4;
3 //Import Statements
4 import java.util.Scanner;
5 //Class Declaration
6 public class Main {
7     // Field Declarations
8     private String name;
9     public static final double PI = 3.14159;
10
11     // Constructor Declaration
12     public Main() {
13         this.name = "Default Name";
14     }
15
16     // Method Declaration
17     public void printName() {
18         System.out.println(name);
19     }
20
21     // Main Method
22     public static void main(String[] args) {
23         // Creating an object of Main
24         Main obj = new Main();
25         obj.printName();
26
27         // Example of using Scanner
28         Scanner input = new Scanner(System.in);
29         System.out.print("Enter the number of gallons: ");
30         double g = input.nextDouble();
31         double lit = 3.785;
32         System.out.println(g + " gallon(s) is equal to " + (g * lit) + " liters.");
33     }
34
35     // Inner Class Declaration (Optional)
36     public class InnerClass {
37         // Inner class body
38     }
39 }
```

2. Describe the difference between upper camel case and lower camel case and provide an example of when you would them.

A. Camel case is a common naming convention in programming where the first letter of each word in a compound word is capitalized, except for the first word in lower camel case. Here's a detailed explanation of the two types:

Upper Camel Case (also known as Pascal case) capitalizes the first letter of each word, including the very first word. This convention is typically used for naming classes and interfaces.

Example:

1. Class Names
2. Interface Names:

Usage:

1. **Classes:** EmployeeDetails, CustomerAccount, MainWindow
2. **Interfaces:** DataProcessor, EventListener, Comparable

Lower Camel Case capitalizes the first letter of each word except the first word, which starts with a lowercase letter. This convention is typically used for naming variables, methods, and sometimes function parameters.

Example:

1. Variable Names
2. Method Names
3. Parameter Names

Usage:

1. Variables: employeeId, firstName, currentBalance

2. Methods: calculateSalary(), getEmployeeDetails(), setCustomerName()

3. Parameters: int accountNumber, String lastName, double interestRate

Summary

- **Upper Camel Case:** First letter of every word is capitalized, used for class and interface names.
 - Examples: EmployeeDetails, DataProcessor
- **Lower Camel Case:** First letter of the first word is lowercase, subsequent words start with an uppercase letter, used for variables, methods, and parameters.
 - Examples: employeeId, calculateSalary(), accountBalance

3. What syntax is used to import the entire Java utilities package? And if you import an entire package do you also need to import additional classes in the same package separately?

A.

Importing the Entire Java Utilities Package

To import the entire Java utilities package, you use the * wildcard character. This tells the Java compiler to import all the public classes and interfaces within that package. The syntax is as follows:

SYNTAX:import java.util.*;

Need for Additional Imports

When you import an entire package using the * wildcard, you don't need to import individual classes from that package separately. For example, if you have:

SYNTAX:import java.util.*;

You can use any class from the java.util package, such as ArrayList, HashMap, Scanner, etc., without additional import statements for those classes.

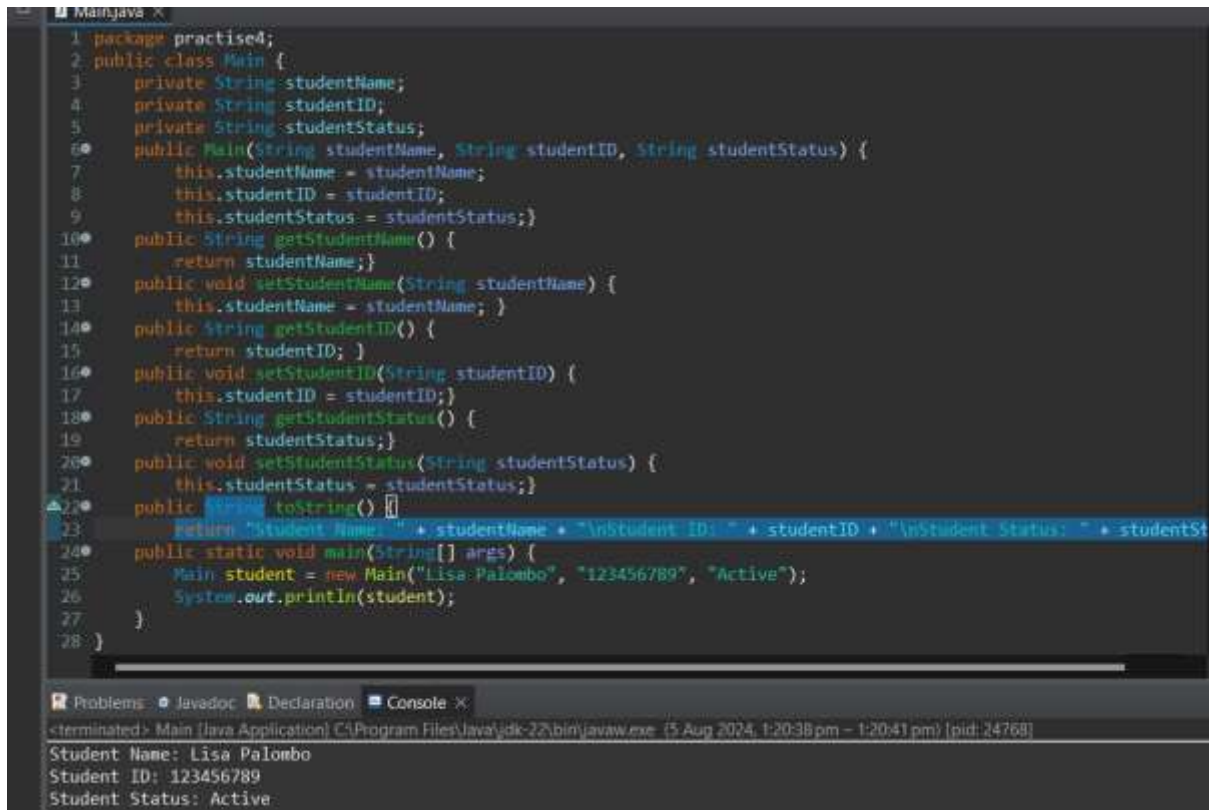
4. Write the syntax for a simple Java object class named Student with the following format:

Student Name: Lisa Palombo

Student ID: 123456789

Student Status: Active.

A.



```
1 package practised4;
2 public class Main {
3     private String studentName;
4     private String studentID;
5     private String studentStatus;
6     public Main(String studentName, String studentID, String studentStatus) {
7         this.studentName = studentName;
8         this.studentID = studentID;
9         this.studentStatus = studentStatus;
10    }
11    public String getStudentName() {
12        return studentName;
13    }
14    public void setStudentName(String studentName) {
15        this.studentName = studentName;
16    }
17    public String getStudentID() {
18        return studentID;
19    }
20    public void setStudentID(String studentID) {
21        this.studentID = studentID;
22    }
23    public String getStudentStatus() {
24        return studentStatus;
25    }
26    public void setStudentStatus(String studentStatus) {
27        this.studentStatus = studentStatus;
28    }
29    public String toString() {
30        return "Student Name: " + studentName + "\nStudent ID: " + studentID + "\nStudent Status: " + studentStatus;
31    }
32    public static void main(String[] args) {
33        Main student = new Main("Lisa Palombo", "123456789", "Active");
34        System.out.println(student);
35    }
36 }
```

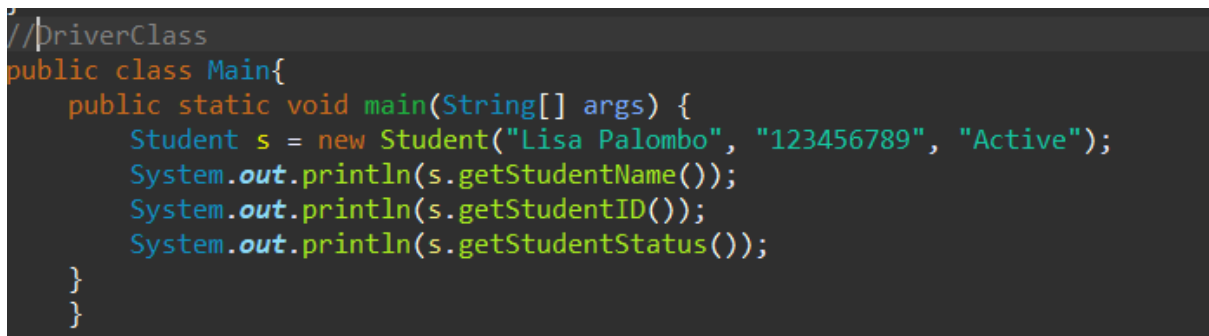
Problems • Javadoc • Declaration • Console ×

<terminated> Main [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (5 Aug 2024, 1:20:38 pm - 1:20:41 pm) [pid:34768]

Student Name: Lisa Palombo
Student ID: 123456789
Student Status: Active

5. Write the code for a Driver Class that will create a Student Object and print the information about the object to the screen.

A.



```
//DriverClass
public class Main{
    public static void main(String[] args) {
        Student s = new Student("Lisa Palombo", "123456789", "Active");
        System.out.println(s.getStudentName());
        System.out.println(s.getStudentID());
        System.out.println(s.getStudentStatus());
    }
}
```

6. From this lesson, list 10 Java keywords.

- A. 1.abstract
- 2.assert
- 3.boolean
- 4.break
- 5.byte

6.case

7.catch

8.char

9.class

10.const

7. Complete the programmer-created object class below. Read the comments for instructions.

A.

```
public class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public static void main(String[] args) {
        Person person = new Person("John Doe", 30);
        System.out.println("Person Name: " + person.getName());
        System.out.println("Person Age: " + person.getAge());
        person.setAge(31);
        System.out.println("Updated Person Age: " + person.getAge());
    }
}
```

8. The program done by using the given comments is

Program:

A.

```
class Person {
    private String name;
    private int age;
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}

public class ManagingPeople {
    public static void main(String[] args) {
        Person p1 = new Person("Arial", 37);
        Person p2 = new Person("Joseph", 15);
        if (p1.getAge() == p2.getAge()) {
            System.out.println(p1.getName() + " is the same age as " + p2.getName());
        } else {
            System.out.println(p1.getName() + " is NOT the same age as " + p2.getName());
        }
    }
}
```