Patuakhali Science and Technology University

Assignment on

"Solve the exercise 3.1 to 3.10"

Course Code: CCE-122
Course Title: Object Oriented Programming

Level - I; Semester - II

Submitted By

Name: M.D. Sakibul Islam Shovon
ID: 2302056
REG: 11834
Session: 2023-2024
Faculty of Computer Science and Engineering

Submitted To

Prof. Dr. Md. Samsuzzaman
Professor of Computer and Communication Engineering Department
Faculty of Computer Science and Engineering

Solve 3.1 to 3.10

3.1 Fill in the blanks:

a) Each class declaration that begins with keyword **public** must be stored in a file that has exactly the same name as the class and ends with the **.java** filename extension.
b) Keyword **class** in a class declaration is followed immediately by the class's name.
c) Keyword **new** requests memory from the system to store an object, then calls the corresponding class's constructor to initialize the object.
d) Each parameter must specify both a(n) **type** and a(n) **name**.
e) By default, classes that are compiled in the same directory are considered to be in the same package, known as the **default package**.
f) Java provides two primitive types for storing floating-point numbers in memory: **float** and **double**.
g) Variables of type **double** represent **double-precision** floating-point numbers.
h) Scanner method **nextDouble** returns a **double** value.
i) Keyword **public** is an access **modifier**.
j) Return type **void** indicates that a method will not return a value.
k) Scanner method **nextLine** reads characters until it encounters a newline character, then returns those characters as a **String**.
l) Class **String** is in package **java.lang.**
m) A(n) **import declaration** is not required if you always refer to a class with its fully qualified class name.
n) A(n) **floating-point number** is a number with a decimal point, such as 7.33, 0.0975, or 1000.12345.
o) Variables of type **float** represent **single-precision** floating-point numbers.
p) The format specifier **%f** is used to output values of type **float** or **double**.
q) Types in Java are divided into two categories—**primitive types** and **reference types**.

3.2 True/False:

a) False – By convention, method names begin with a *lowercase* first letter, and subsequent words use uppercase.
b) True – import is not needed for classes in the same package.
c) True – Empty parentheses mean no parameters are required.
d) False – Primitive variables cannot invoke methods; object references are needed.
e) False – Local variables are only accessible within their declared method.
f) True – Method bodies are enclosed in {}.
g) False – Primitive *local* variables are *not* initialized by default (unlike instance

variables).
h) True – Reference-type instance variables default to null.
i) True – Any class with public static void main(String[] args) can execute an app.
j) True – Argument count must match parameter count.
k) False – Floating-point literals are double by default (use f suffix for float).

3.3 Difference between local and instance variables:

Answer: A *local variable* is declared inside a method and is only accessible within that method. An *instance variable* is declared in a class (outside methods) and is accessible to all methods of the class.

3.4 Purpose of a method parameter vs. argument:

Answer: parameter represents additional information that a method requires to perform its task. Each parameter required by a method is specified in the method's declaration. An argument is the actual value for a method parameter. When a method is called, the argument values are passed to the corresponding parameters of the method so that it can perform its task.

3.5

(Keyword new) What's the purpose of keyword new? Explain what happens when you use it.

Answer:
The keyword new is used to:

1. Allocate memory for a new object.
2. Call the constructor of the class to initialize the object.

Example:

```
MyClass obj = new MyClass();
```

- new MyClass() → Allocates memory and calls the constructor.
- The reference (obj) stores the memory address of the newly created object.

3.6

(Default Constructors) A class declares a constructor that takes two parameters. How would you create an instance of the class with no parameters?

Answer:
If a class defines any constructor (e.g., MyClass(int a, int b)), Java does not provide a default no-argument constructor. To create an instance with no parameters:

1. Add a no-argument constructor explicitly:

   public MyClass() { }

2. Then instantiate:
   MyClass obj = new MyClass();

Otherwise, compilation fails because new MyClass() won't match any existing constructor.

3.7

(Instance Variables) Explain the purpose of an instance variable.

Answer:

- Instance variables hold the state of an object.
- They are declared inside a class but outside any method.
- Each object gets its own copy of these variables.
- Example:

```
class Car {
   String color;  // Instance variable
}
```

3.8

(Instance Variables) Explain the purpose of an instance variable. (Using Classes without Importing Them) Most classes need to be imported before they can be used in an app. Why is every app allowed to use classes System and String without first importing them?

Answer:

- String and System are part of the java.lang package.
- java.lang is automatically imported in every Java program.
- No explicit import is needed for classes in this package.

3.9

(Using a Class without Importing It) Explain how a program could use class Scanner without importing it.

Answer:
Use the fully qualified name (including the package):

Example:

```
java.util.Scanner input = new java.util.Scanner(System.in);
```

Avoids import java.util.Scanner; but makes code verbose.

3.10

(set and get Methods) Explain the disadvantage of creating a class that has no set and get methods for an instance variable.

Answer:
Disadvantages:

1. No Control Over Access: Without get/set, variables must be declared public, exposing them to unrestricted modification.

Example:

```
public class Person {
  public String name;  // Bad: Direct access allows invalid values.
}
```

2. No Validation: set methods can enforce rules (e.g., if (age > 0) this.age = age;).
3. Breaks Encapsulation: Violates OOP principles by allowing external code to modify internal state directly.

Solution:
Always use private variables with public get/set methods:

```
public class Person {
  private String name;
  public String getName() { return name; }
  public void setName(String name) { this.name = name; }
}
```