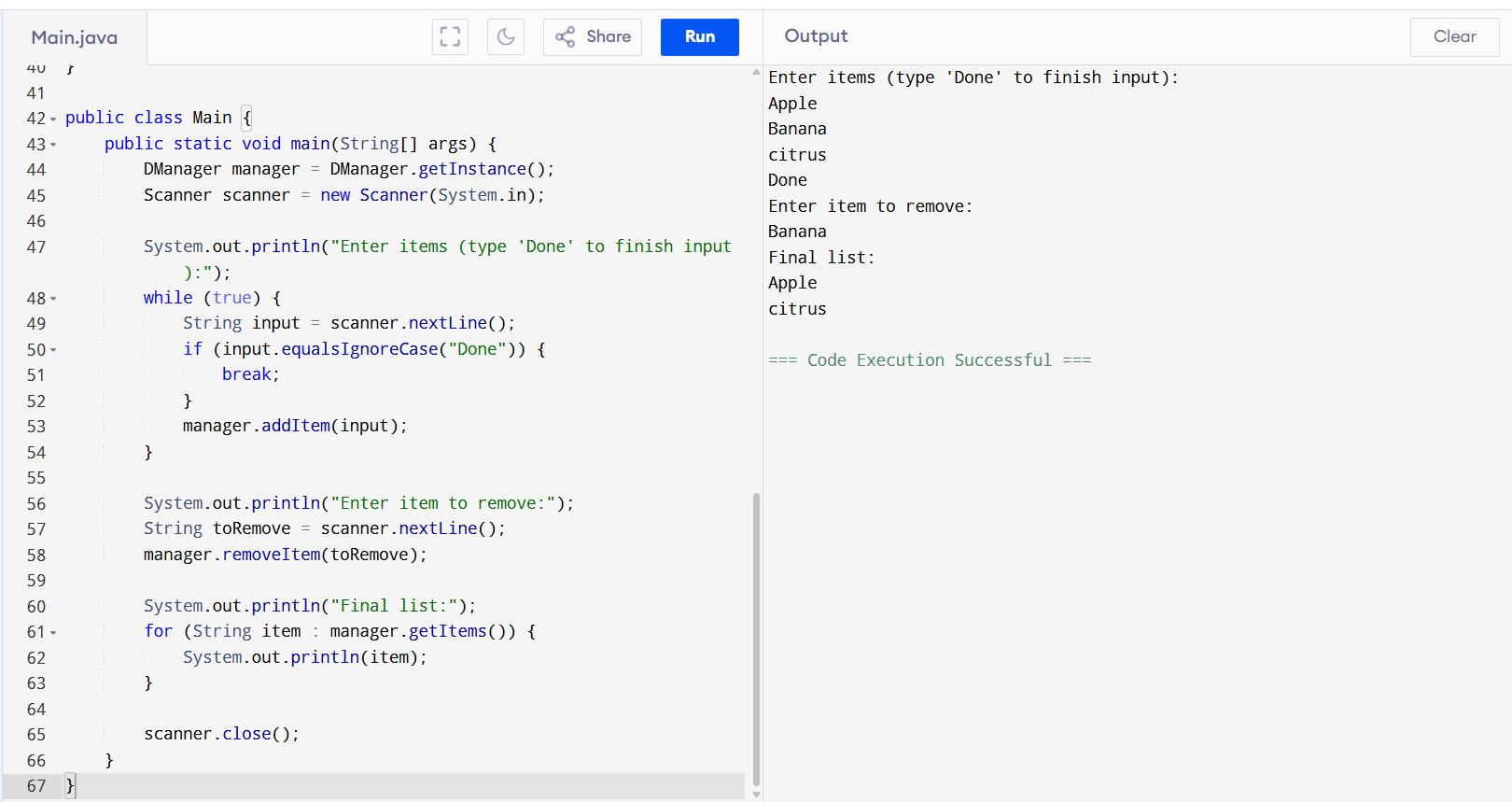
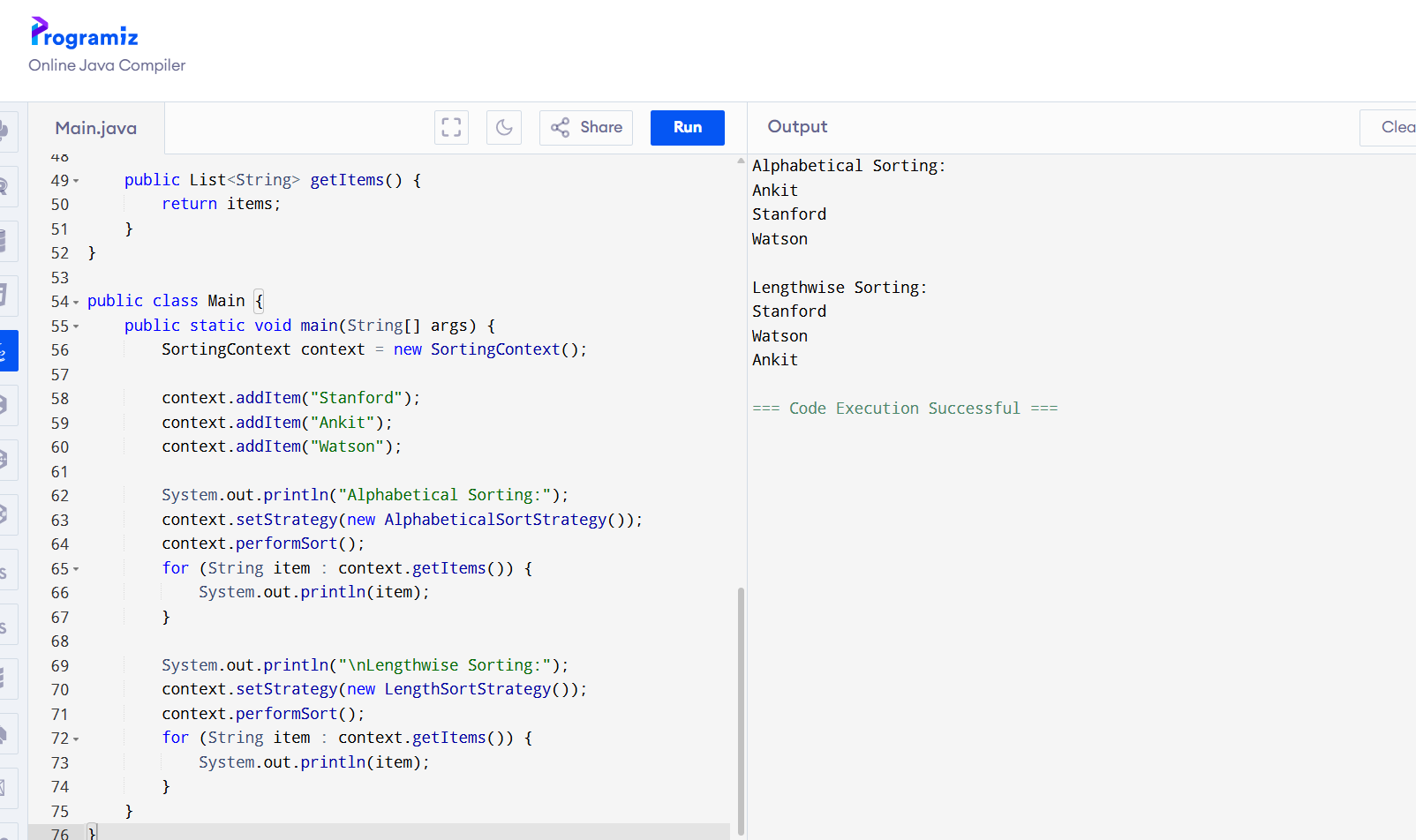
Task:1



Task2:



Task3:

Class A {

psvm() {

Int a = 5;

Int b = 10;

Int c = 15:

sout((a > b ) && (b < c))

}

}

1. Compilation error
2. True
3. False
4. Runtime error

Task 4:

Finding Inheritance during requirement analysis is it important in OOAD .. why so?

1. It removes the need for encapsulation in the system design
2. It helps identify objects with the shared behavior to promote code reuse and logical hierarchy
3. It forces a flat class design improving performance by reducing polymorphic calls
4. Ensures all classes are instantiated using interfaces.

Task 5:

Which characteristics best defines polymorphism in OOP?

1. Ensures each class has its own copy of data members
2. It restricts method access to specific roles within a system
3. It allows a single function or operator to behave differently based on its parameters or calling object
4. It serialized different objects into a common file format for persistence

Task 6:

Which of the following best explains the concept of data hiding in Object-Oriented Programming?

1. Data hiding means removing data from memory when no longer in use to ensure memory efficiency.
2. Data hiding involves using access specifiers to restrict direct access to class members, enabling controlled interaction through methods.
3. Data hiding refers to storing object data in secure databases during runtime.
4. Data hiding is achieved by deleting unused attributes from objects after object creation.

Task 7:

 In OOAD, what is the primary value of Requirements Analysis?

1. It helps define class inheritance structure before testing

2. It identifies system behavior and user needs to model objects and interactions meaningfully

 3. It configures application deployment scripts for testing

4.It automatically generates interface documentation from class files

Task 12:

Why is refactoring considered a continuous part of modern software development?

1. Refactoring is performed only at the end of a release cycle for documentation purposes

2. It replaces traditional debugging with automatic patching mechanisms

1. Continuous refactoring ensures that the design evolves with changing requirements, reducing technical debt and improving code health