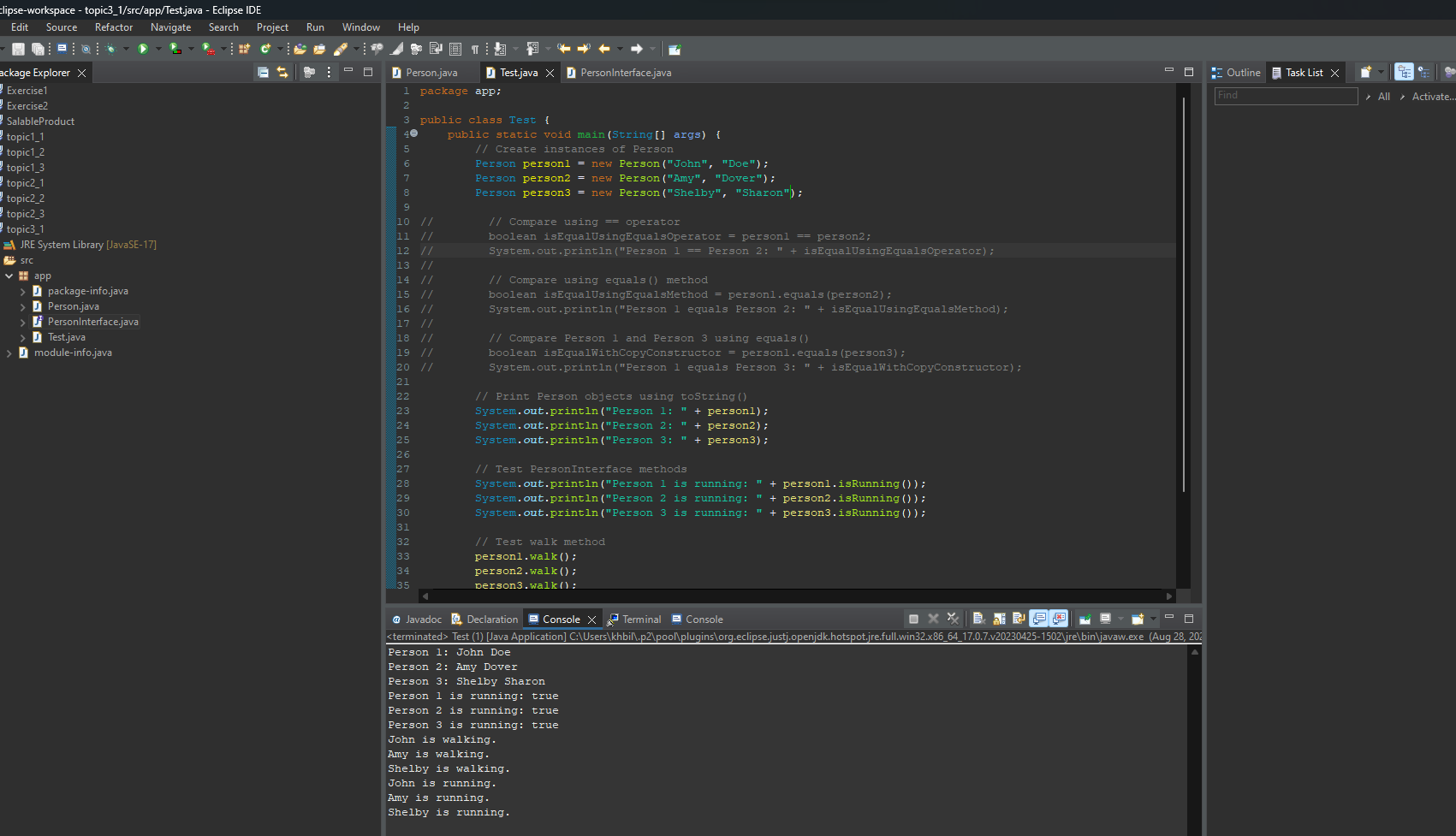
Topic 3

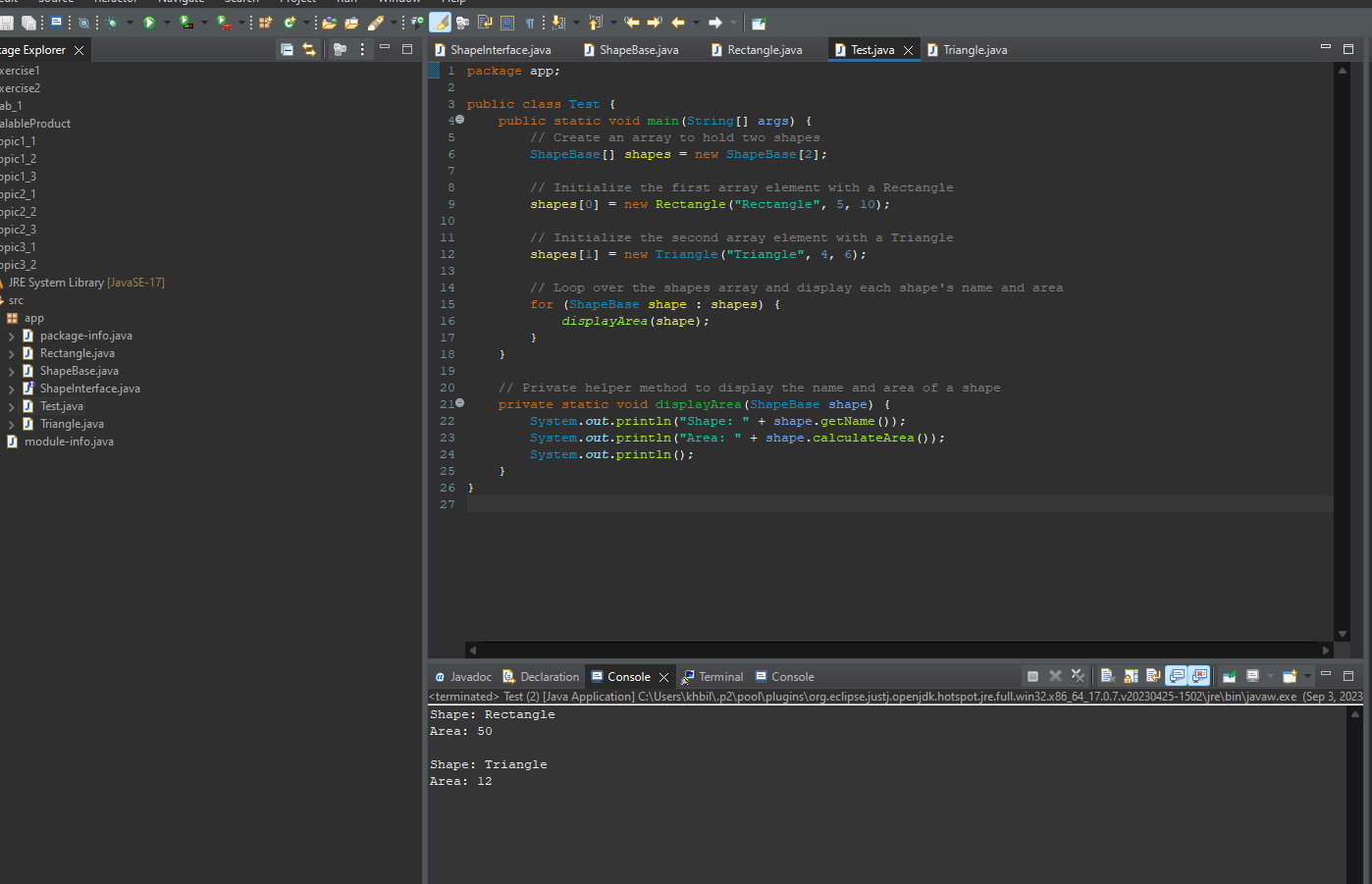
KariAnn Harjo

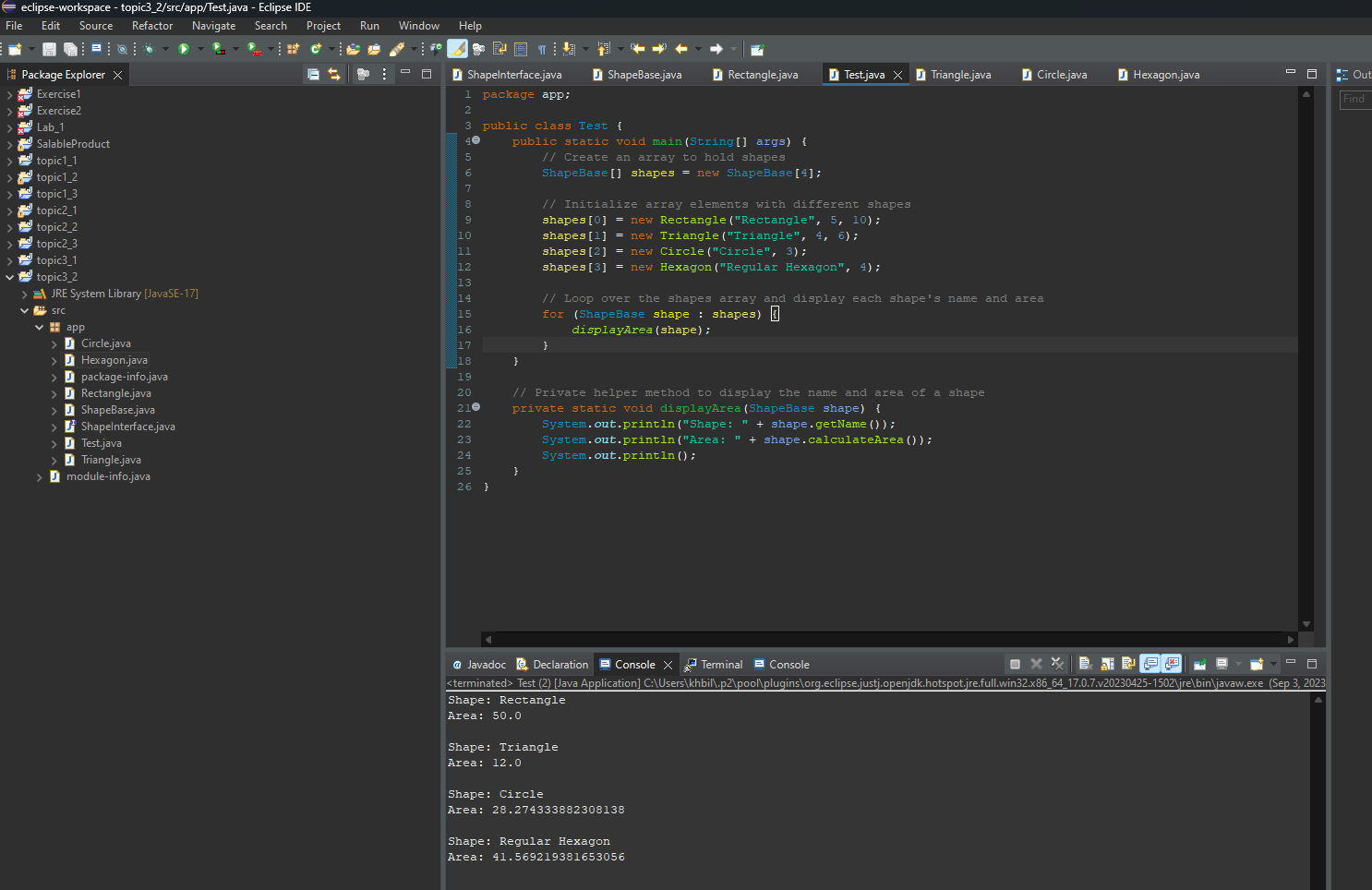
09/03/2023

Part 1: Person Interface

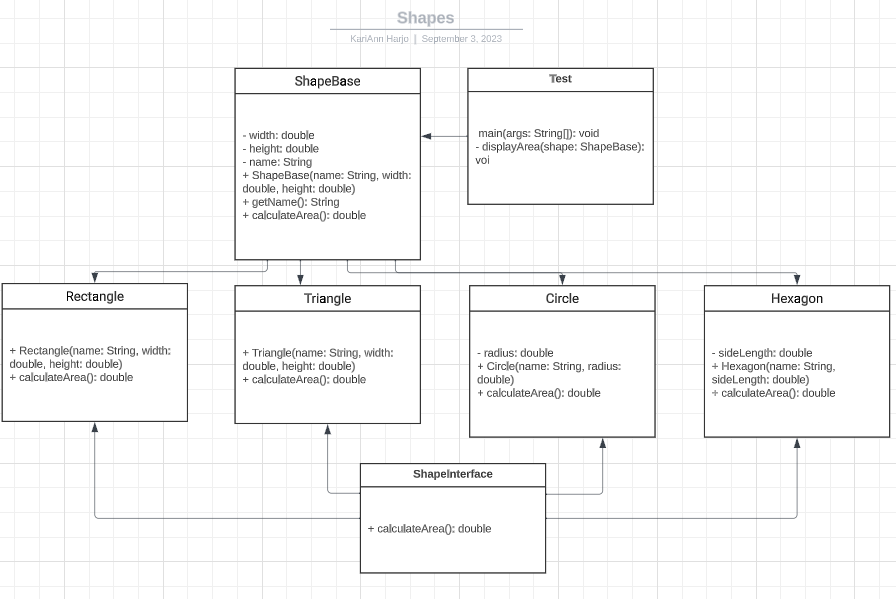


Part 2: Shape

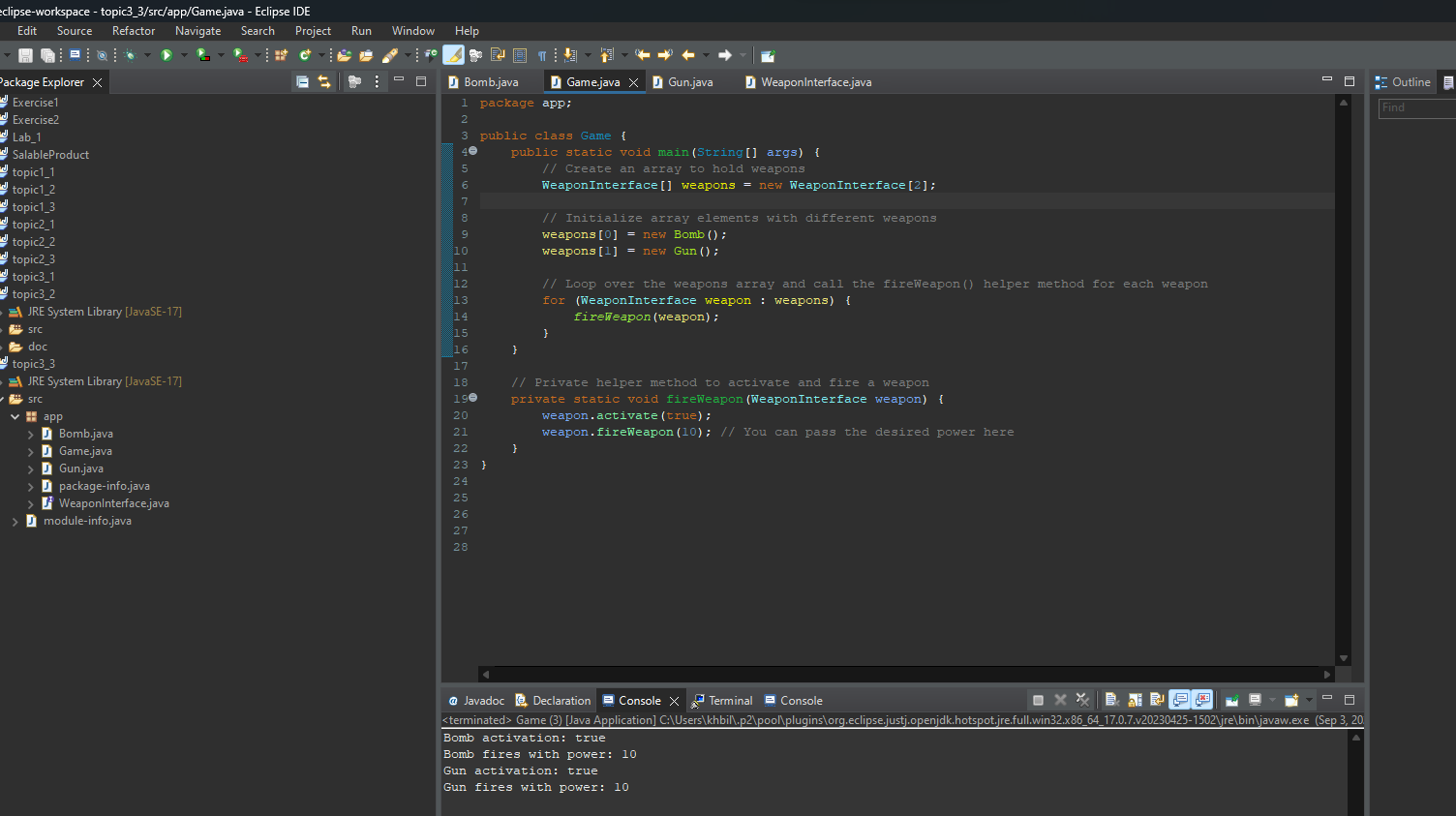




Write-Up  
Polymorphism is demonstrated in the provided code through the use of inheritance and method overriding. Specifically, the `ShapeBase` class implements the `ShapeInterface` interface and defines a method called `calculateArea()`. Subclasses such as `Rectangle`, `Triangle`, `Circle`, and `Hexagon` override this method with their own implementations to calculate the area specific to their shape. In the `Test` class, an array of `ShapeBase` objects is used to hold instances of these subclasses, and the `calculateArea()` method is called on each of them without needing to know their specific types. This showcases polymorphism, as the same method name (`calculateArea()`) is used on objects of different classes, and the appropriate implementation is automatically selected at runtime based on the object's actual class.

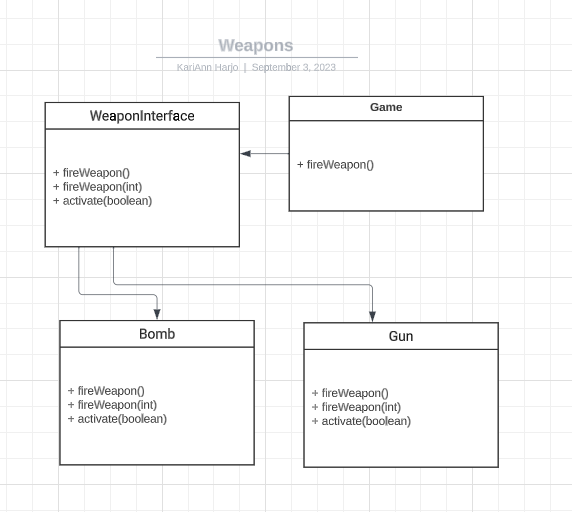


**Part 3: Weapons**



Write-Up

Polymorphism is demonstrated in the provided code through the use of interfaces and method overriding. Specifically, the `WeaponInterface` serves as a common contract for the `Bomb` and `Gun` classes, allowing objects of both classes to be referenced by the interface type. This enables polymorphic behavior when the `fireWeapon()` and `activate()` methods are called on objects of these classes in the `Game` class. The `Game` class can invoke these methods without knowing the specific implementation details of each weapon class, showcasing polymorphism in action.



**Part 4: Debugging**

