Part 1: 15 True/False (1 point each)

Question **1**
Complete
Points out of 1.00
Flag question

Unit tests combine tests for multiple classes.

Select one:
- True
- False

Question **2**
Complete
Points out of 1.00
Flag question

Inheritance allows you to remove public methods from the child class that the parent class has.

Select one:
- True
- False

Question **3**
Complete
Points out of 1.00
Flag question

Concrete factories in the Factory Method pattern can be implemented using generics.

Select one:
- True
- False

Question **4**
Complete
Points out of 1.00
Flag question

An interface is an abstract class but not all abstract classes are interfaces.

Select one:
- True
- False

Question 5
Complete
Points out of 1.00
Flag question

Abstract factories usually have methods to create multiple different kinds of abstract (instead of concrete) products.

Select one:
- True
- False

Question 6
Complete
Points out of 1.00
Flag question

We are allowed to inherit a method in a child class but change it from private to public.

Select one:
- True
- False

Question 7
Complete
Points out of 1.00
Flag question

An interface can be instantiated with `new`.

Select one:
- True
- False

Question 8
Complete
Points out of 1.00
Flag question

The Strategy pattern requires the use of multiple inheritance.

Select one:
- True
- False

Question 9
Complete
Points out of 1.00
Flag question

Refactoring a method changes its result values for a given input.

Select one:
- True
- False

Question 10
Complete
Points out of 1.00
Flag question

Factory methods should have `void` return types.

Select one:
- True
- False

Question 11
Complete
Points out of 1.00
Flag question

In the Strategy pattern, a class is allowed to have more than one strategy.

Select one:
- True
- False

Question 12
Complete
Points out of 1.00
Flag question

Builder methods return "this" to allow method chaining.

Select one:
- True
- False

Question 13
Complete
Points out of 1.00
Flag question

Achieving 100% tests passed guarantees that there are no bugs in the program.

Select one:
- True
- False

Question 14
Complete
Points out of 1.00
Flag question

The thing inside the angle brackets in Iterator<String> is a generic type parameter.

Select one:
- True
- False

Question 15
Complete
Points out of 1.00
Flag question

A State is allowed to modify the object that contains it (e.g., to swap to a different state).

Select one:
- True
- False

Part 2: 10 Multiple Choice (3 points each)

Question 1

Complete

Points out of 3.00

Flag question

In the dependency relationship shown here, which of the following cannot be a true statement:

Select one:

- a. Class B is a local variable for a method of A
- b. Class B is an input parameter for a method of A
- c. Class A has a declared instance of variable B
- d. Class B has a declared instance of variable A

Question 2

Complete

Points out of 3.00

Flag question

Which of these assertions will fail?

Select one:

- a. `assertTrue( 2 + 2 == 4 )`
- b. `assertFalse( 2 + 2 == 4 )`
- c. `assertNotEqual( 2 + 2, 5 )`
- d. `assertEqual( 2 + 2, 4 )`

Question 3

Complete

Points out of 3.00

Flag question

In OOD, what is the relationship where Y is either an input parameter to a method in X or Y is local to a method in X?

Select one:

- a. Association
- b. Aggregation
- c. Dependency
- d. Composition

Question **4**

Complete

Points out of 3.00

Flag question

Which pattern can be used to cleanly replace a Visitor?

Select one:

- a. Factory method
- b. Anti-visitor
- c. Iterator
- d. Singleton

Question **5**

Complete

Points out of 3.00

Flag question

Test-driven development is

Select one:

- a. compiling and debugging code in small increments.
- b. validating the product (system under development) using customer feedback every step in the process lifecycle.
- c. when test code is written first, followed by writing the corresponding production code.

Question **6**

Complete

Points out of 3.00

Flag question

Which of the following implements the `getInstance()` method of a singleton named `HardwareManager`.

Select one:

a.
```
public HardwareManager getInstance() {
    if( instance == null )
        instance = new HardwareManager();
    return new HardwareManager();
}
```

b.
```
public static void getInstance() {
    instance = new HardwareManager();
}
```

c.
```
public static HardwareManager getInstance() {
    if( instance == null )
        instance = new HardwareManager();
    return instance;
}
```

d.
```
public static HardwareManager getInstance() {
    if( instance == null )
        return null;
    return new HardwareManager();
}
```

Question **7**

Complete

Points out of 3.00

Flag question

What is Duck Typing?

Select one:

- a. It is the use of an abstract class in the Alice programming language.
- b. Some programming languages allow untyped function parameters, as long as those objects have identical formal parameter lists for methods needed.
- c. It is equivalent to the concept of casting in Java.
- d. It is a Perl programming language data type.

Question 8

Complete

Points out of 3.00

Flag question

Which kind of test coverage is defined by every Boolean expression being evaluated as both true and false?

Select one:

- a. Statement coverage
- b. Condition coverage
- c. Expression coverage
- d. Branch coverage
- e. Function coverage

Question 9

Complete

Points out of 3.00

Flag question

A code smell

Select one:

- a. is code that stays in sync.
- b. may refer to poorly functioning team dynamics.
- c. is a slang term for a failed unit test.
- d. may refer to duplicated code.

Question 10

Complete

Points out of 3.00

Flag question

What is the UML Class Diagram notation to denote multiplicity of 1 or more?

Select one:

- a. 1++
- b. *
- c. 1..*
- d. 1 or more

Part 3: 7 Short Answer (5 points each)

Question 1
Complete
Points out of 5.00
Flag question

What are the 4 principles of object oriented programming?

abstraction, encapsulation, inheritance, polymorphism

Question 2
Complete
Points out of 5.00
Flag question

Complete the following Java singleton by implementing `getInstance()`, using the correct return type, visibility, method attributes, and body:

```
class VideoSystem {
    private GlContext context;
    private ScreenBuffer[2] buffers;
    private static VideoSystem system;

    //put your getInstance() method here:

}
```

```
public static VideoSystem getInstance() {
if (instance == null)
instance = new VideoSystem();
return instance;
}
```

Question 3
Complete
Points out of 5.00
Flag question

What is it called when adding new code causes your old tests to fail?

Answer: regression

Question 4
Complete
Points out of 5.00
Flag question

Which kind of testing is performed on a single module, method, or class?

Answer: unit testing

Question 5
Complete
Points out of 5.00
Flag question

Which kind of testing combines multiple modules, methods, or classes?

Answer: integration tests

Question 6
Complete
Points out of 5.00
Flag question

What is the primary difference between the Abstract Factory and Factory Method patterns (remember, the factory method pattern often also has an abstract class in it, so don't say "one is abstract")? (1-2 sentences)

The Factory Method pattern creates specific products, while the Abstract Factory Method pattern creates a family of related and replaceable products.

Question 7
Complete
Points out of 5.00
Flag question

How do components communicate with each other?

Answer: interfaces

Part 4: 1 Long Answer (10 points)

Question **1**

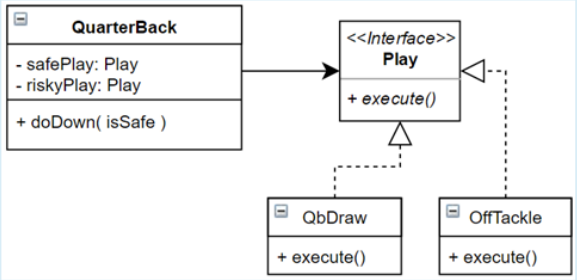
Complete

Points out of 10.00

Flag question

Given the following UML, write the code for each class and interface in Java or C++. The `doDown()` method should check its argument: if it's true, it should execute `safePlay`, otherwise `riskyPlay`. For each concrete class, implement "`execute()`" to print "`Executing: [name of the play]`" (replace brackets with actual name). Ensure correct return types and visibility.

What pattern is this?



This is the command pattern.

```
public interface Play {
 public void execute();
}

public class QuarterBack {
 private Play safePlay;
 private Play riskyPlay;
 public QuarterBack(Play safePlay, Play riskyPlay) {
  this.safePlay = safePlay;
  this.riskyPlay = riskyPlay;
 }
 public void doDown(isSafe) {
  if(isSafe) safePlay.execute();
  else riskyPlay.execute();
 }
}

public class QbDraw implements Play {
 public void execute() {
  System.out.println("Executing: QbDraw");
 }
}

public class OffTackle implements Play {
 public void execute() {
  System.out.println("Executing: OffTackle");
 }
}
```

Part 5: 1 Long Answer (10 points)

Question **1**

Complete

Points out of 10.00

Flag question

Write **EXACTLY** enough JUnit tests to achieve complete statement coverage of the `foo` method. Excess tests will be penalized. Use a single assert_equals for each test, correct return types, and attributes.

```
public class Bork {
    public static int foo( int bar ) {
        if( bar % 3 == 0 )
            return -40;
        if( bar < 11 )
            return 200;
        return 500;
    }
}
```

```
class BorkTest {
 @Test
 void testReturnNegativeForty() {
  Bork bork = new Bork();
  assertEquals( bork.foo(3), -40 );
 }

 @Test
 void testReturnTwoHundred() {
  Bork bork = new Bork();
  assertEquals( bork.foo(7), 200 );
 }

 @Test
 void testReturnFiveHundred() {
  Bork bork = new Bork();
  assertEquals( bork.foo(50), 500 );
 }
}
```

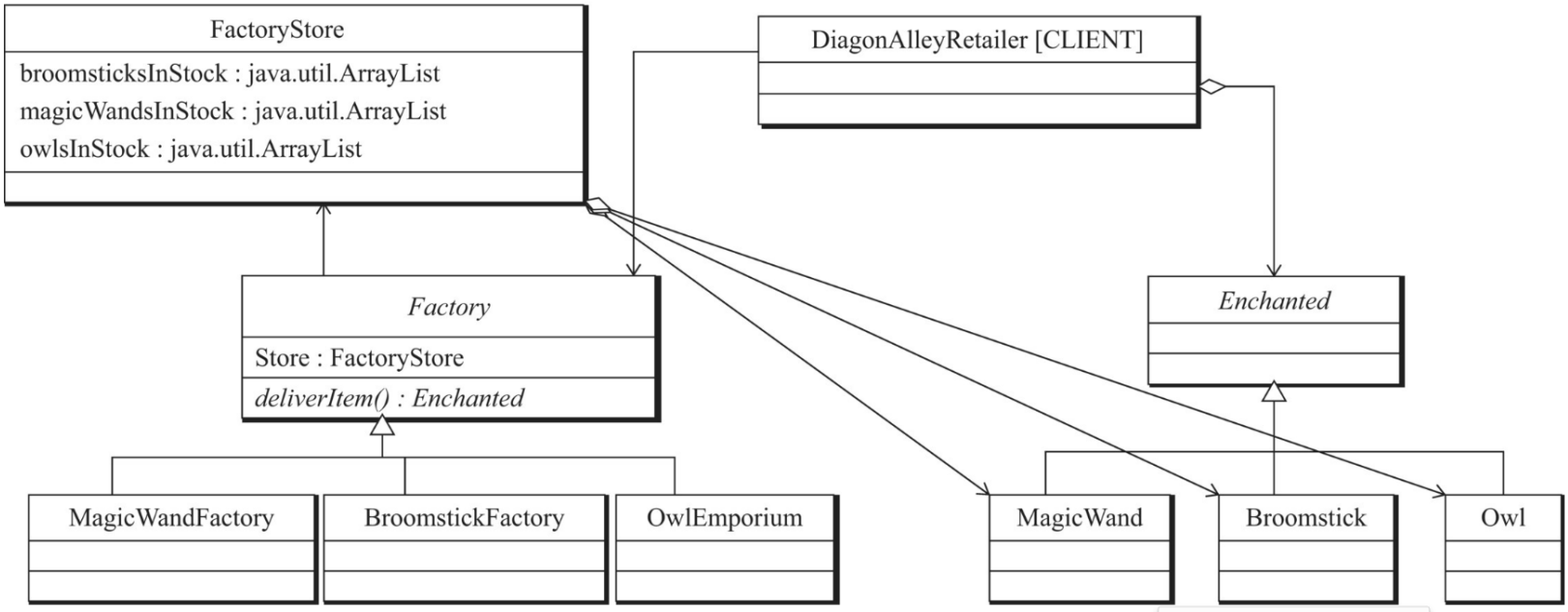
Note:

Here are the following possible choices for all future pattern identification problems.

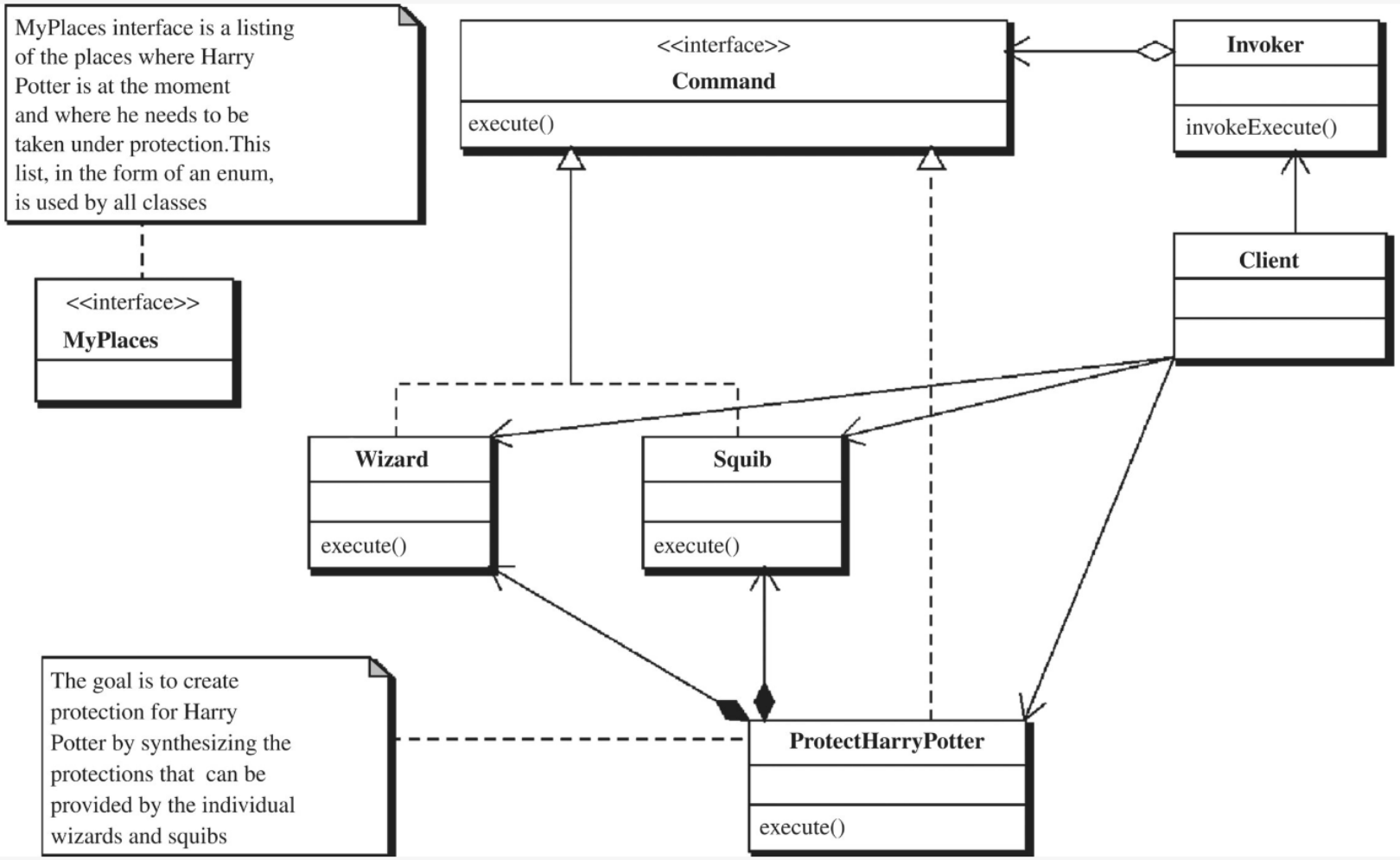
Select one:

- a. The State Pattern
- b. The Visitor Pattern
- c. The Composite Pattern
- d. The Iterator Pattern
- e. The Observer Pattern
- f. The Factory Method
- g. The Strategy Pattern
- h. The Builder Pattern
- i. The Command Pattern
- j. The Adapter Pattern
- k. The Abstract Factory Pattern
- l. The Singleton Pattern

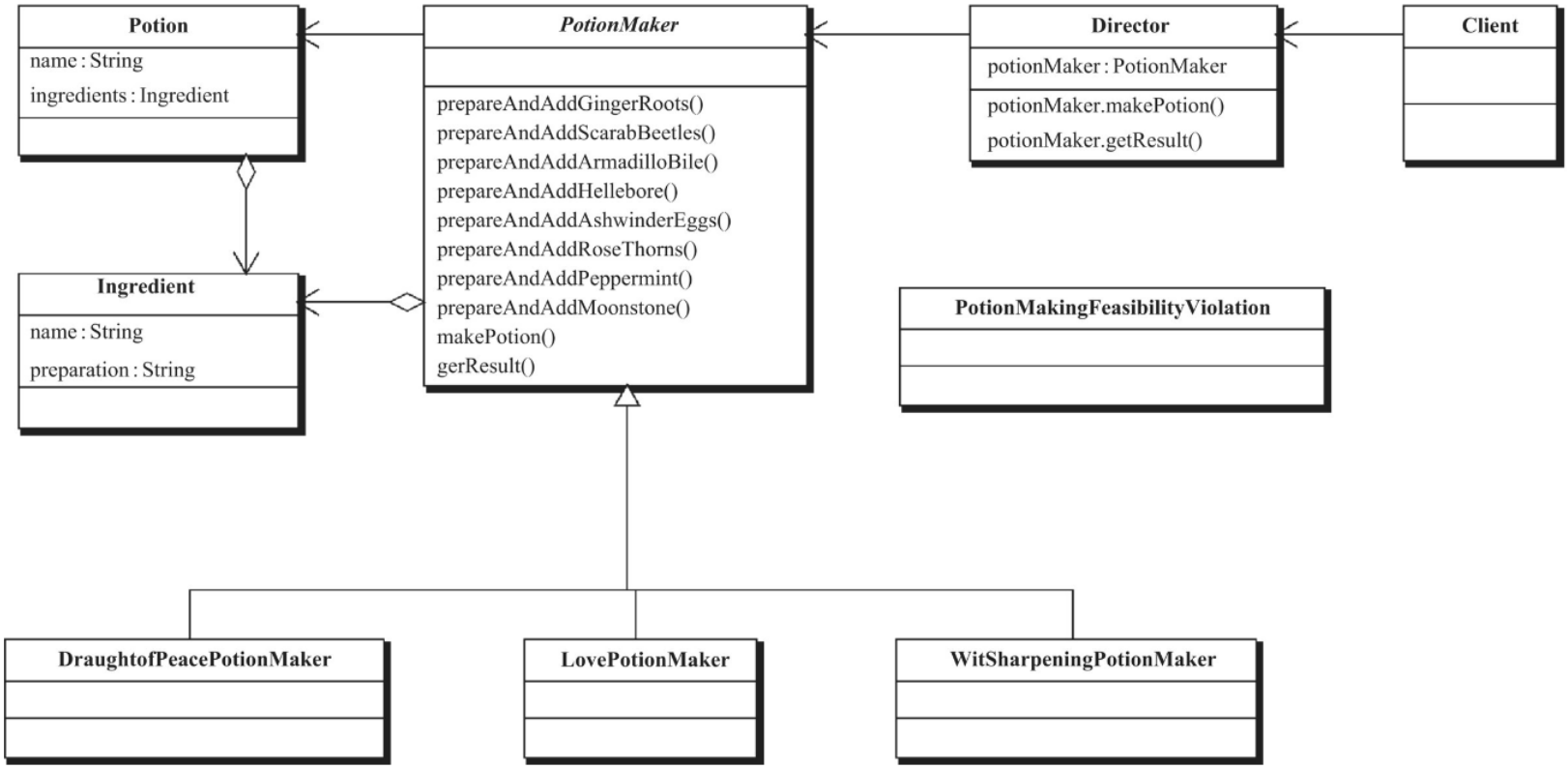
Part 6: 6 Pattern Identifications (3 points each)



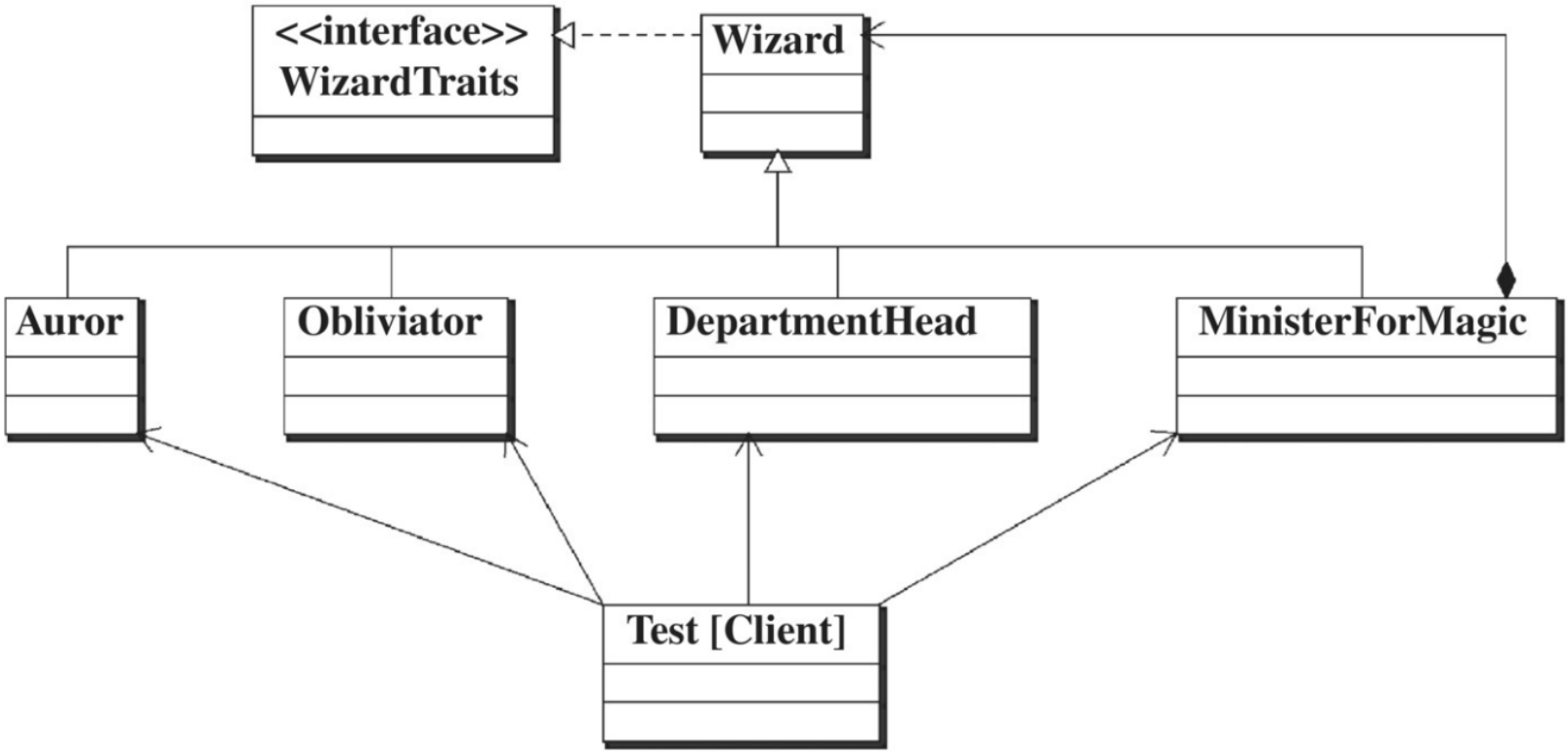
I picked Factory.



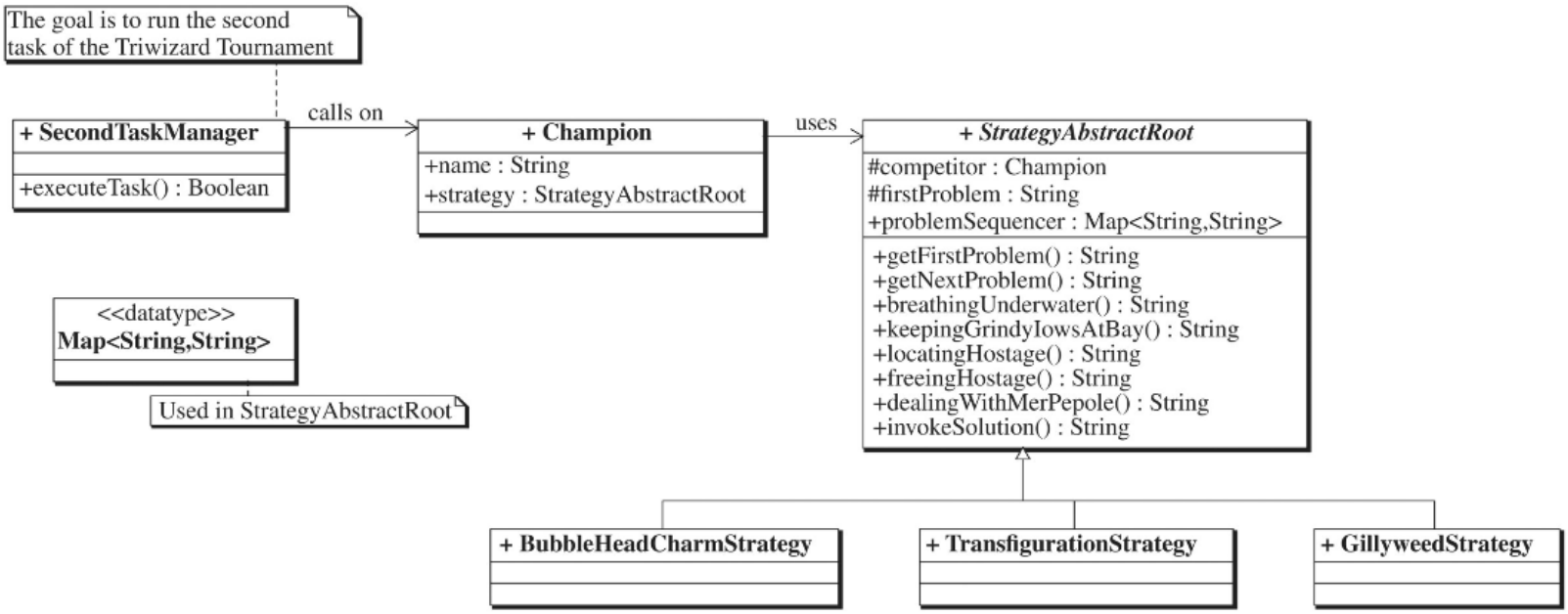
I picked Command.



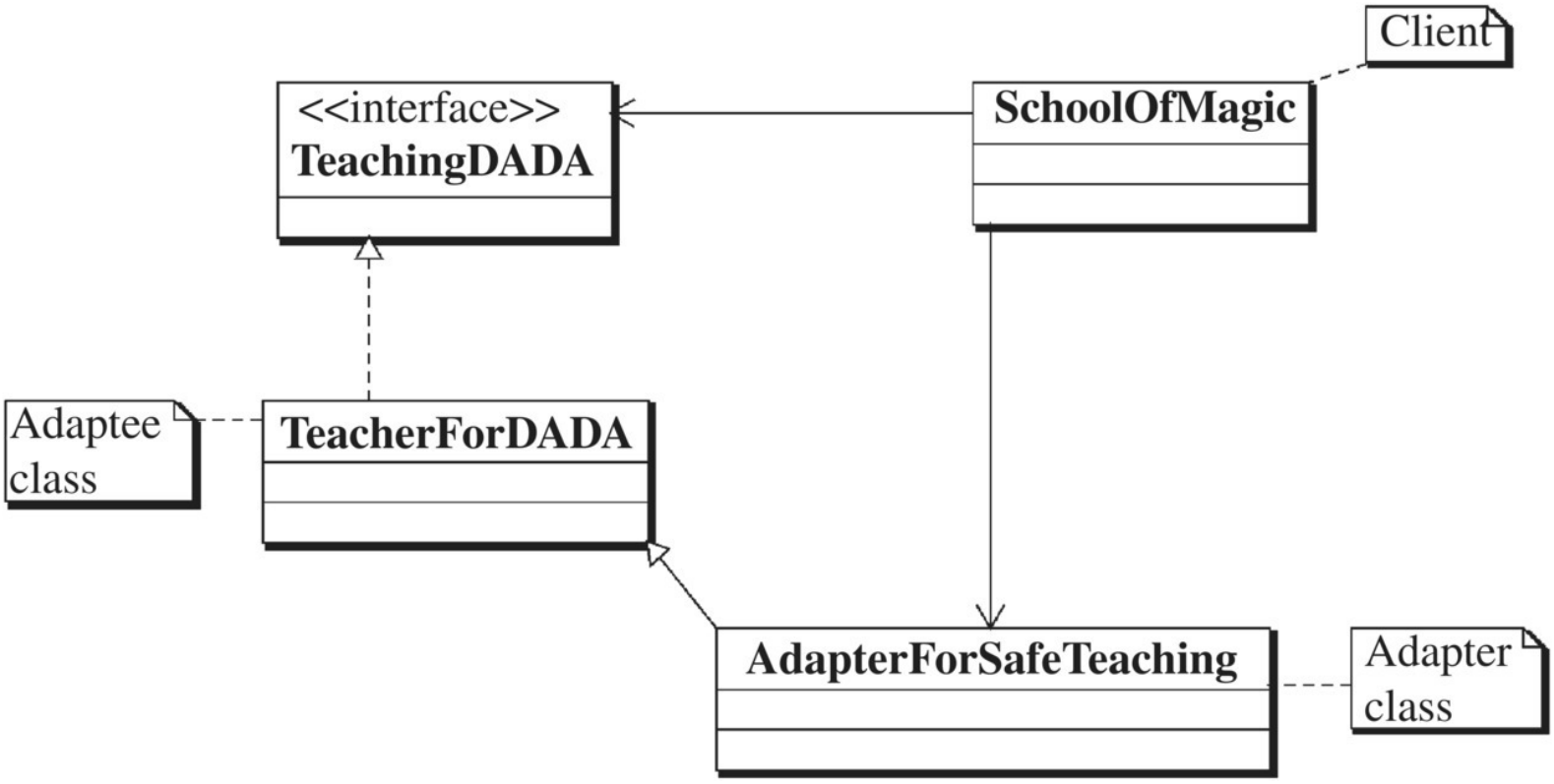
I picked Factory.



I picked Strategy.

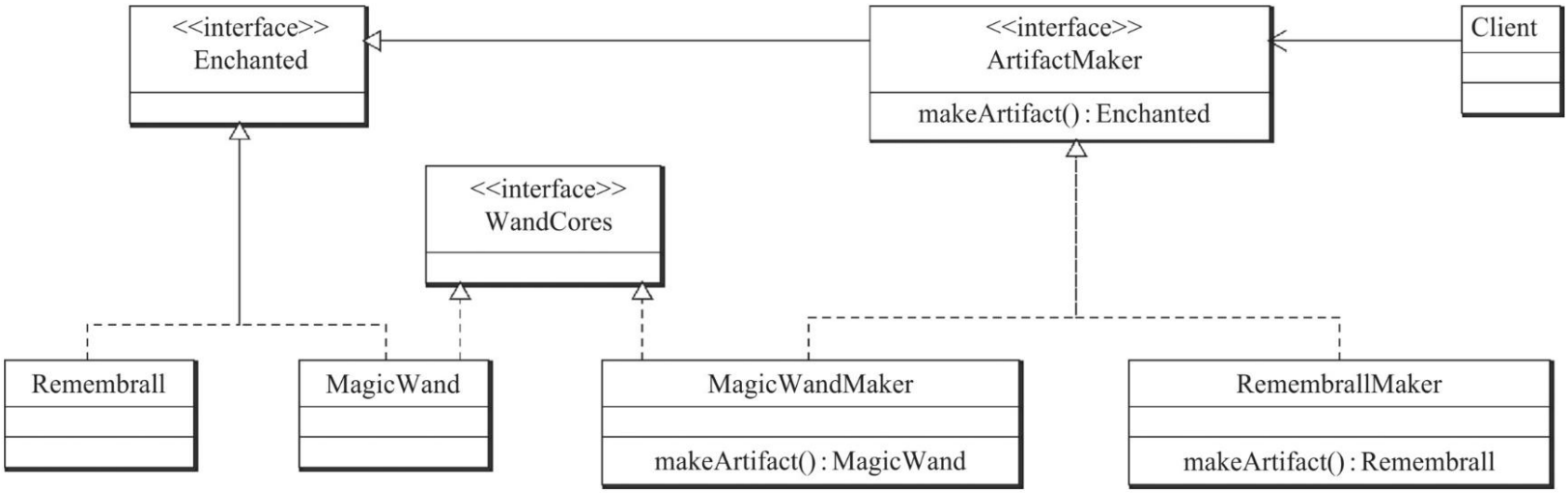


I picked Strategy.

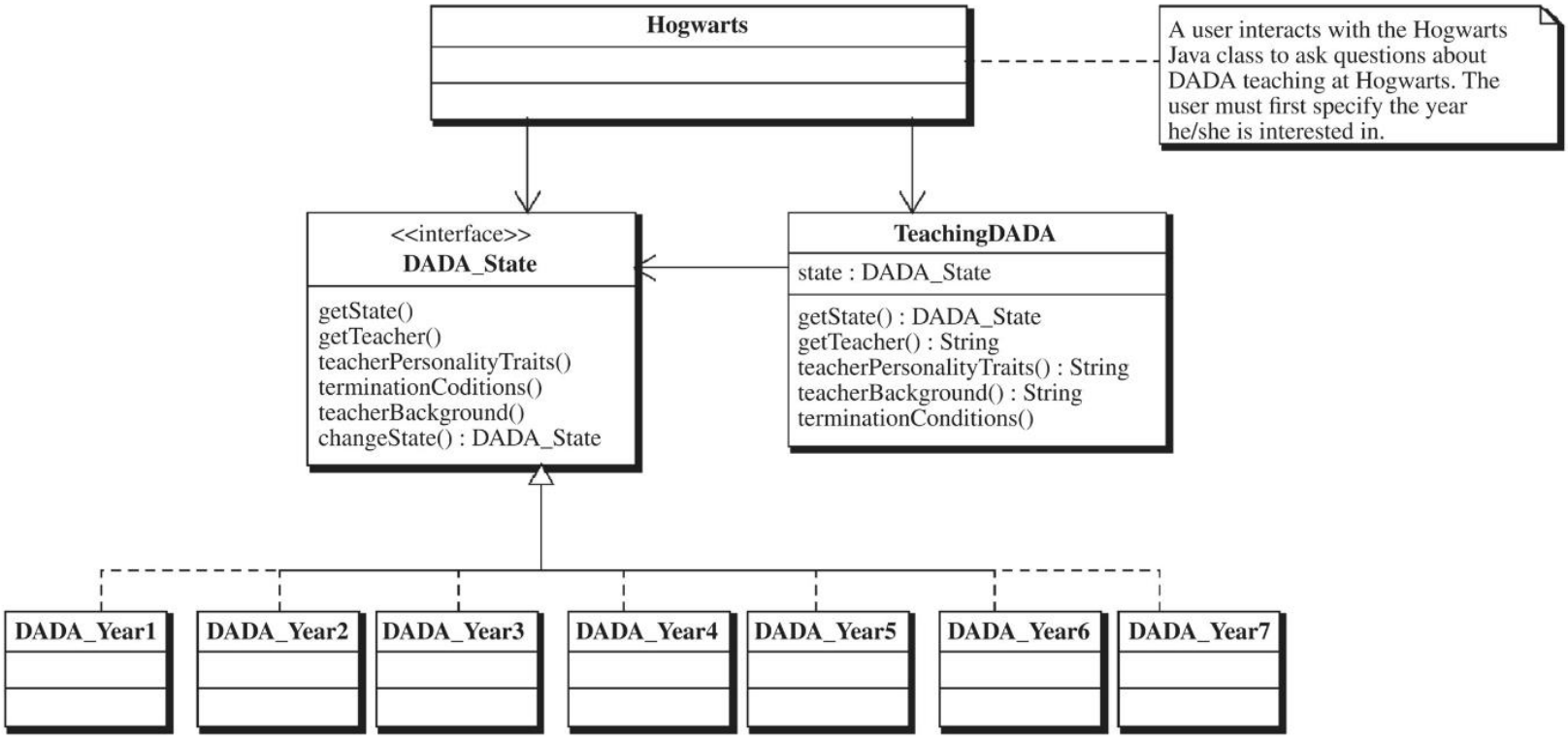


I picked Adapter.

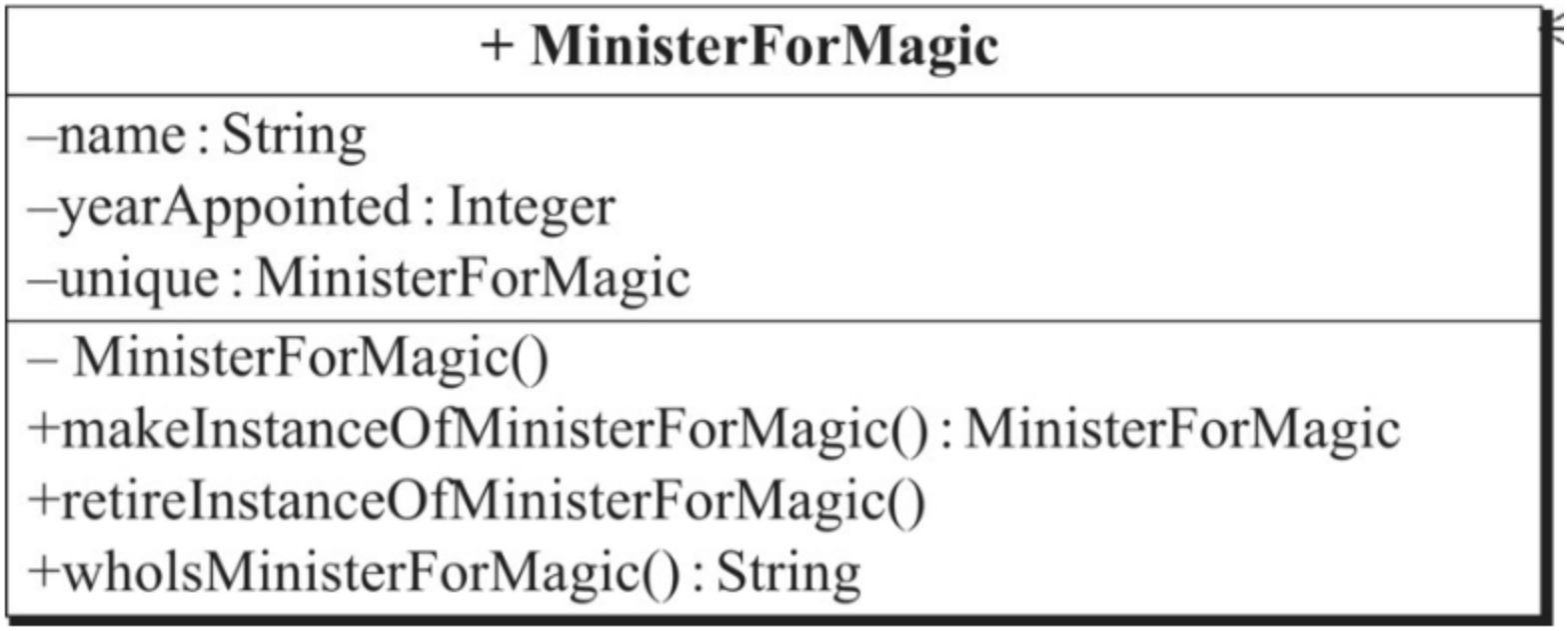
Part 7: 6 Pattern Identifications (3 points each)



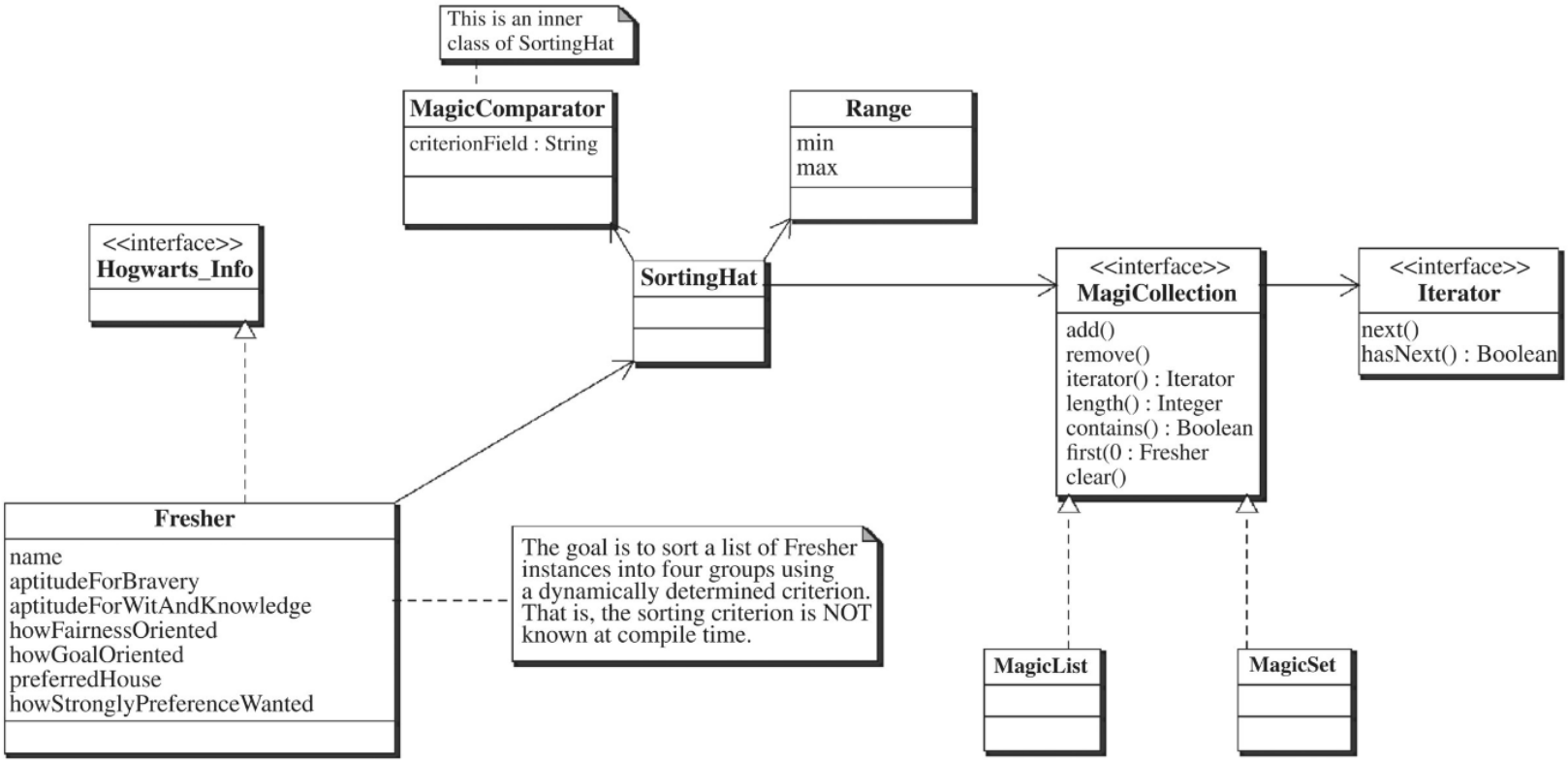
I picked Factory.



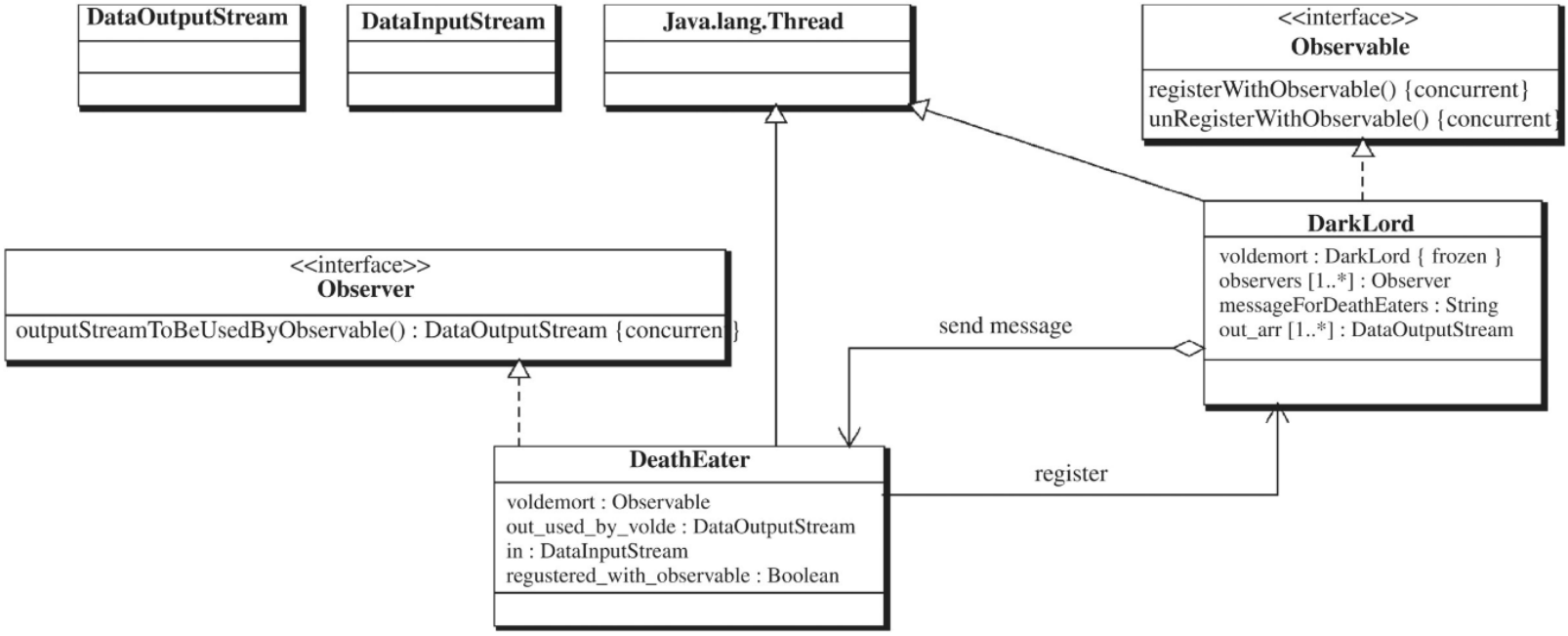
I picked Builder.



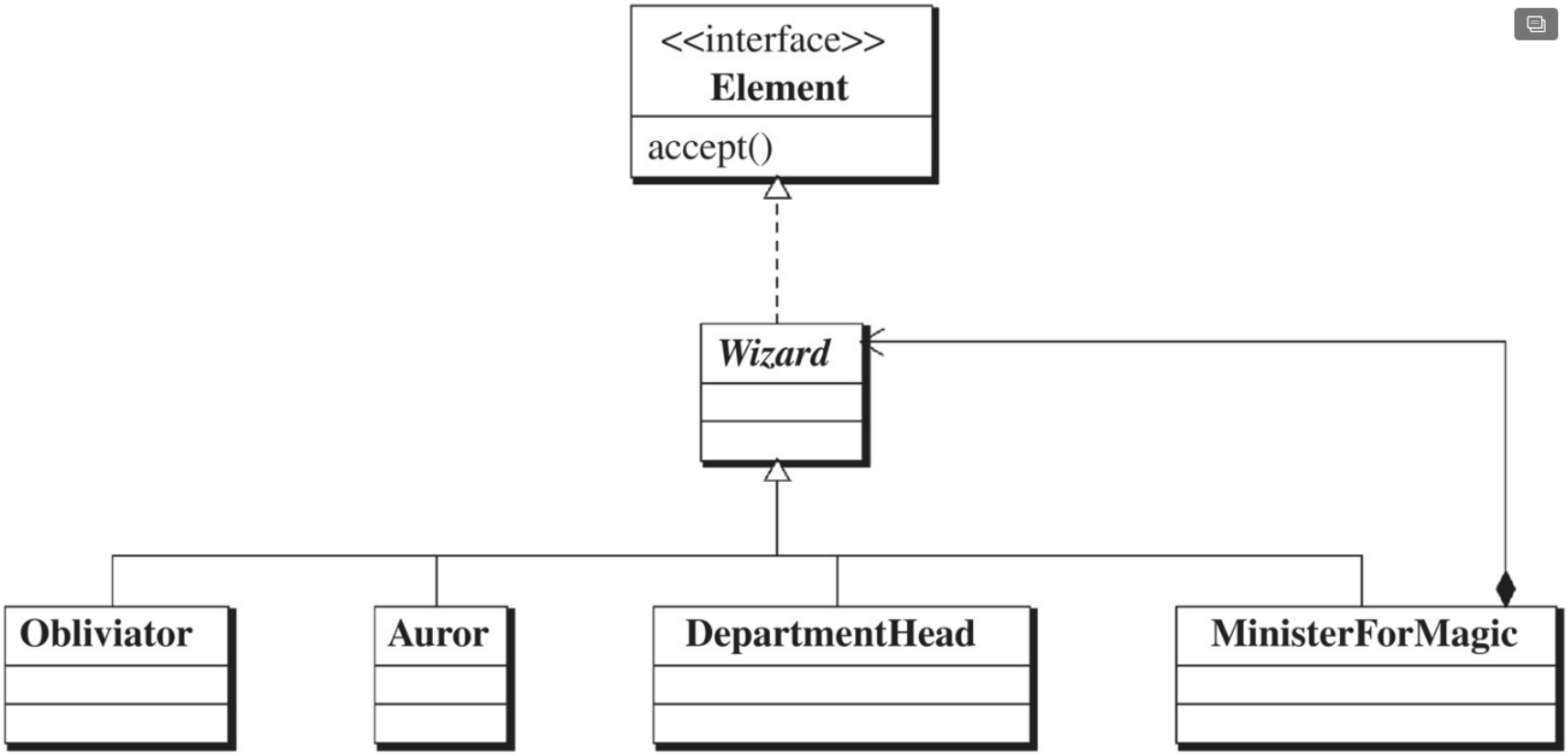
I picked Singleton.



I picked Iterator.



I picked Observer.



I picked Visitor.