Final 2 Module A

Due Dec 16 at 11:59pm **Points** 25 **Questions** 25
Time Limit 35 Minutes

Instructions

These question include:

- UML Class Diagrams (focus on meaning of relationship arrows)
- Functional Programming (focus on handout)
- Design Patterns (focus on slides and HFDP handout)
- Design of Everyday Things (focus on slides)
- A few project-related questions from projects 2 and 4

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	34 minutes	19 out of 25

Score for this quiz: **19** out of 25
Submitted Dec 14 at 5:58pm
This attempt took 34 minutes.

Question 1 — 1 / 1 pts

During its life, software will undergo change.

Correct!

- (•) True
- ( ) False

Question 2 — 1 / 1 pts

An association arrow between classes A and B in a UML class diagram (see below) corresponds most closely to which of the following in Java.

- A implements B
- A has a field of type B

Correct!

- A imports B
- A extends B

Question 3

1 / 1 pts

A dotted generalization arrow between classes A and B in a UML class diagram (see below) corresponds most closely to which of the following in Java.

- A has a field of type B
- A has multiple elements of type B

Correct!

- A implements B
- A extends B

Question 4

1 / 1 pts

Which design pattern allows objects to be notified when another object's state changes?

Correct!

- (selected) Observer
- Decorator
- Strategy
- Adapter

Question 5

1 / 1 pts

Which design pattern encapsulates the creation of an object and allows it to be constructed in steps

Correct!

- (selected) Builder
- Observer
- Factory Method
- Decorator

Question 6

1 / 1 pts

Which of the following is NOT a design concern if condiments are represented as booleans in an abstract Beverage class that calculates condiment cost?

Changing a condiment's price forces us to alter existing code

Correct!

Will not be able to calculate cost for a beverage with more than one condiment

All condiments may not make sense for all beverages

Adding a new condiment to the design will force us to alter code in the cost method

Question 7

0 / 1 pts

Which Java package or framework uses the Observer pattern to monitor events?

ou Answered

The Java I/O package

The Java Concurrency framework

The Java Collections framework

orrect Answer

The Java GUI framework (Swing)

Question 8

1 / 1 pts

Which of the following patterns is NOT a behavioral pattern?

Observer

Strategy

Correct!

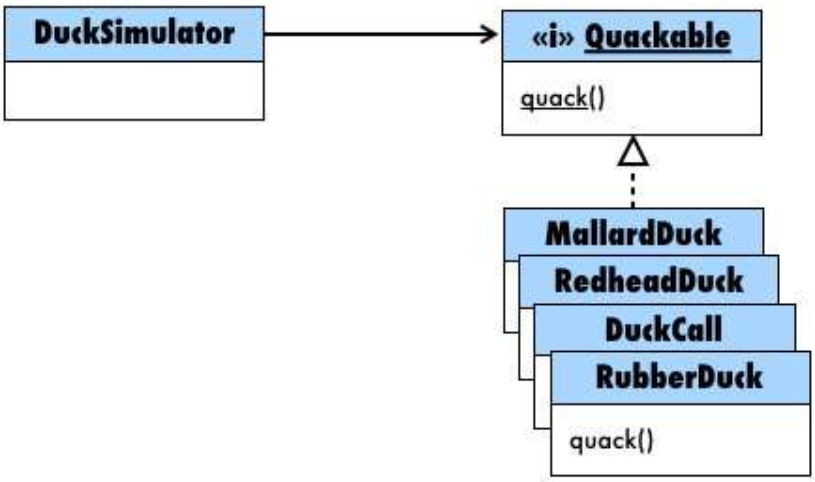
Factory

Iterator

Question 9

1 / 1 pts

In the duck simulator design given below, what design pattern would we apply if we wanted the ability to count the number of quacks from each duck?



Observer

Correct!

Decorator

Factory

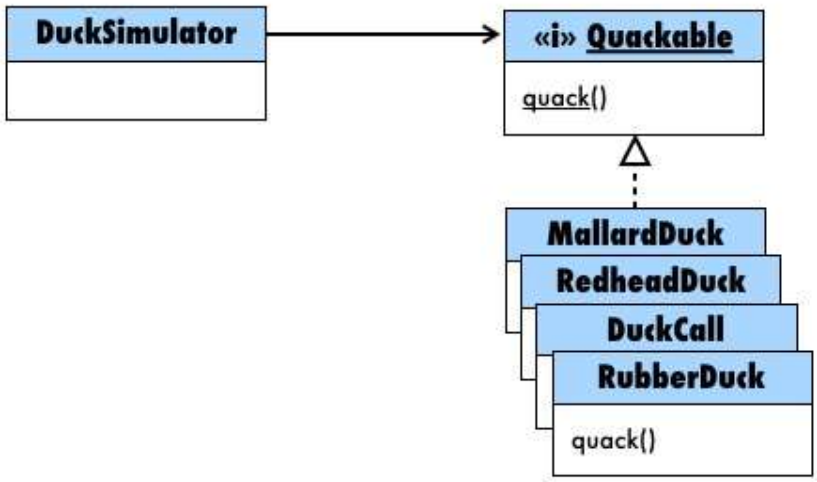
Adapter

Composite

Question 10

0 / 1 pts

In the duck simulator design given below, we would not need to significantly modify existing code if we added the Composite pattern



orrect Answer

- True

ou Answered

- False

Question 11

1 / 1 pts

Identify the aspects of your application that vary and separate them from what stays the same. What design goal does this echo?

- Low Coupling

Correct!

- Encapsulation

- Abstraction

- High Cohesion

Question 12

1 / 1 pts

The Open-Closed design principle states that existing code should be extended but not modified

Correct!

- True
- False

Question 13

1 / 1 pts

Which pattern violates the single responsibility principle?

Correct!

- Strategy
- Composite
- Iterator
- Adapter

Question 14

1 / 1 pts

In The Design of Everyday Things, Interaction Design is focused on

- Abstraction and Decomposition

- Emotional Impact
- Form and Material

Correct!

- Understandability and Usability

Question 15

1 / 1 pts

In The Design of Everyday Things, which of the following is NOT related to the Fundamental Principles of Design?

Correct!

- Encapsulation
- Affordances
- Signifiers
- Conceptual Modal

Question 16

0 / 1 pts

In The Design of Everyday Things, the barbs on a barbed-wire fence are an example of this

- A false affordance
- Both a false affordance and a signifier
- A hidden affordance
- A signifier

ou Answered

- An anti-affordance

orrect Answer

- Both an anti-affordance and a signifier

Question 17

1 / 1 pts

In The Design of Everyday Things, a traditional door knob is an example of this

- An anti-affordance
- A false signifier

Correct!

- An affordance for turning
- An invisible signifier

Question 18

1 / 1 pts

In the design of a Lego motorcycle, once you have placed the red and yellow lights, there is only one place to put the blue lights. This is an example of

- Physical constraint

Correct!

- Logical constraint
- Cultural constraint
- Semantic constraint

Question 19

1 / 1 pts

In Project 4 (DuckSim), which of the following best describes the relationship of the Bling class (the duck decorator) to the Duck class.

- Bling has a field of type Duck
- Bling inherits from Duck
- Bling both inherits from Duck and has a Duck field

Correct!

- Bling neither inherits from Duck nor does it have a Duck field

Question 20

1 / 1 pts

Consider the following incomplete Scala method, which returns the last element in a Scala List.

```
def last[T] (list: List[T]): T = list match {
  case Nil => ...
  case x :: Nil => ...
  case _ :: xs => ...
}
```

Which case is matched if you pass in List("hello")?

- none of the cases are matched

Correct!

- x :: Nil
- _ :: xs
- Nil

Question 21

1 / 1 pts

Consider the following incomplete Scala method, which returns the last element in a Scala List.

```
def last[T] (list: List[T]): T = list match {
  case Nil => ...
  case x :: Nil => ...
  case _ :: xs => ...
}
```

What is the type of xs?

Correct!

- (selected) List[T]
- String
- T
- List[String]

Question 22

1 / 1 pts

Consider the following incomplete Scala method, which returns the last element in a Scala List.

```
def last[T] (list: List[T]): T = list match {
  case Nil => ...
  case x :: Nil => ...
  case _ :: xs => ...
}
```

What should be returned when the last case (_ :: xs) is matched?

Correct!

- (selected) last(xs)
- _

xs

x

Question 23

0 / 1 pts

Consider the following incomplete Scala method, which returns the last element in a Scala List.

```
def last[T] (list: List[T]): T = list match {
  case Nil => ...
  case x :: Nil => ...
  case _ :: xs => ...
}
```

Which case is equivalent to: xs.length == 0?

_ :: xs

orrect Answer

Nil

none of the cases are equivalent to: xs.length == 0

ou Answered

x :: Nil

Question 24

0 / 1 pts

[Project 2]

What is the *representation* of a CircArrayPipe after the following calls?

```
Pipe<String> pipe = new CircArrayPipe<>(2); // Note: capacity of 2!
pipe.append("X");
pipe.append("Y");
```

```
pipe.removeFirst();
pipe.append("Z");
```

- contents = [X, Y] and first = 1 and last = 2 and length = 2
- ou Answered: contents = [Y, Z] and first = 0 and last = 1 and length = 2
- contents = [X, Y, Z] and first = 0 and last = 2 and length = 2
- orrect Answer: contents = [Z, Y] and first = 1 and last = 0 and length = 2

Question 25

0 / 1 pts

Pure functional languages rely heavily on assignment

- ou Answered: True
- orrect Answer: False

Quiz Score: **19** out of 25