Final 2 Module B

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	35 minutes	24 out of 25

Score for this quiz: **24** out of 25
Submitted Dec 14 at 6:37pm
This attempt took 35 minutes.

Question 1 — 1 / 1 pts

A composition 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 one or more elements of type B that continue to exist even if an object of type A is destroyed

Correct!

A has one or more elements of type B that are destroyed when an object of type A is destroyed

A implements B

A extends B

Question 2

1 / 1 pts

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

Correct!

A implements B

A has a field (instance variable) whose type is B

A extends B

A imports B

Question 3

1 / 1 pts

Which design pattern allows users to treat groups of objects the same as they treat individual objects?

- Singleton
- Builder
- Iterator
- Composite

Correct!

Question 4

1 / 1 pts

Which design pattern allows subclasses to decide which concrete classes to create?

- Composite
- Decorator
- Builder
- Factory Method

Correct!

Question 5

1 / 1 pts

Which of the following is NOT a design concern if we notify subscriber objects by calling them sequentially from a publisher object, as in the following code.

```
public void notifyDucks() {
    mallardDuck.update();
    rubberDuck.update();
```

```
    decoyDuck.update();
}
```

- We are coding to concrete implementations rather than interfaces
- If we add a new subscriber object, we have to alter code
- We have no way to add or remove a new subscriber at runtime
- The subscribers do not implement a common interface

Correct!

Question 6

1 / 1 pts

Which Java package or framework implements its own iterator pattern?

- The Java Concurrency framework
- The Java Collections framework
- The Java GUI framework (Swing)
- The Java I/O package

Correct!

Question 7

1 / 1 pts

Which of the following patterns is a creational pattern?

- State
- Composite

Iterator

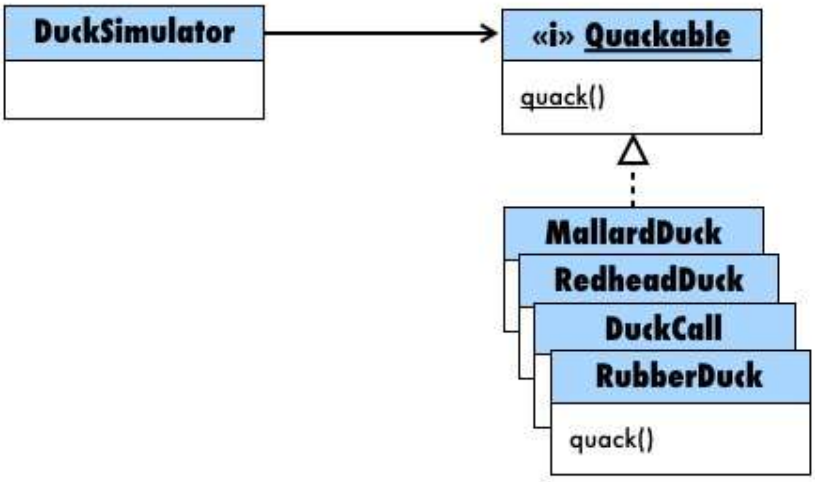
Correct!

Builder

Question 8

1 / 1 pts

In the duck simulator design illustrated below, what design pattern would we apply if we wanted to look at the quacking behavior of families of ducks?



Adapter

Factory

Observer

Correct!

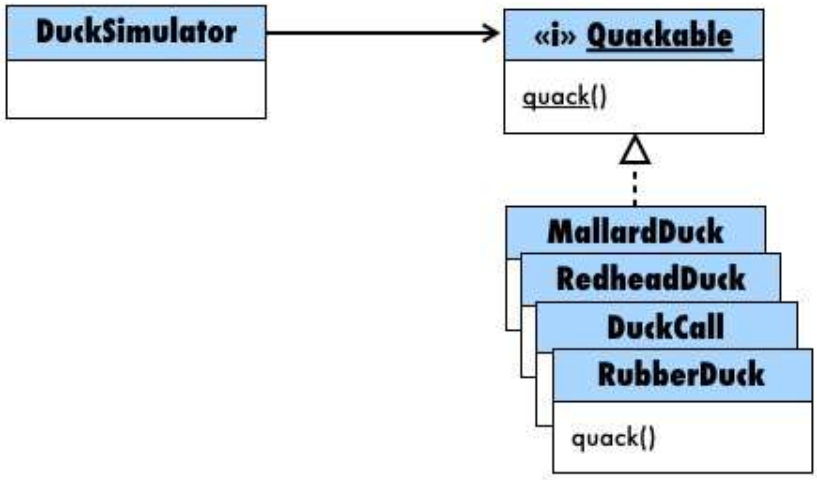
Composite

Decorator

Question 9

1 / 1 pts

In the duck simulator design illustrated below, we would not need to significantly modify existing code if we added a Decorator pattern



Correct!

- (selected) True
- False

Question 10

1 / 1 pts

Program to an interface, not to an implementation. What design goal does this echo?

Correct!

- (selected) Abstraction
- Encapsulation
- Low Coupling
- High Cohesion

Question 11

1 / 1 pts

Implementations change less than interfaces

- True
- False

Correct!

Question 12

0 / 1 pts

Which of the following patterns is a class pattern (rather than an object pattern)?

orrect Answer

- Adapter

ou Answered

- Decorator
- Composite
- Iterator

Question 13

1 / 1 pts

It The Design of Everyday Things, Experience Design is focused on

Correct!

- Emotional Impact

- Form and Material
- Understandability and Usability
- Discoverability and Feedback

Question 14

1 / 1 pts

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

Correct!

- (selected) Cohesion
- Discoverability
- Constraints
- Mappings

Question 15

1 / 1 pts

In The Design of Everyday Things, a high-voltage wire is an example of this

- A false affordance
- A hidden affordance
- An invisible signifier

Correct!

- (selected) A hidden anti-affordance

Question 16

1 / 1 pts

In the Design of Everyday Things, an affordance is a property of an object that exists independently of the individual that uses the object

- True
- False

Correct!

Question 17

1 / 1 pts

In the design of a Lego motorcycle, the lights are interchangeable, but most people know to put the red lights in the back. This is an example of

- Logical constraint
- Cultural constraint
- Physical constraint
- Semantic constraint

Correct!

Question 18

1 / 1 pts

In Project 4 (DuckSim), which code *directly* calls the observer's "update" method?

- The joinDSWC method in the Duck class
- The update method in the Duck class (it is recursive)

Correct!

- (selected) The notifyObserver method in the Observable class
- The createDuck method in the DuckFactory class

Question 19

1 / 1 pts

In Project 4 (DuckSim), the DuckFactory needs to be a singleton class. Can we do this by making DuckFactory an enum type?

- No, enum types cannot be singletons
- Yes, Observable is an interface, so DuckFactory can implement Observable and extend Enum

Correct!

- (selected) No, DuckFactory must extend Observable and therefore it cannot extend Enum
- Yes, Enum is an interface, so DuckFactory can implement Enum and extend Observable

Question 20

1 / 1 pts

Pure functional languages do not allow mutable objects

Correct!

- (selected) True
- False

Question 21

1 / 1 pts

Consider the following Scala list:

```
val list = "This" :: "is" :: "not" :: "a" :: "test" :: Nil
```

What would list.head followed by list.tail return?

- This followed by List(is, not, a, test, Nil)
- This followed by List(is, not, a, test)
- This followed by test
- List(This, is, not, a) followed by test

Correct!

Question 22

1 / 1 pts

Consider the following Scala list:

```
val list = "This" :: "is" :: "not" :: "a" :: "test" :: Nil
```

What would list.drop(2) return?

- List(This, is, not, a, test)
- List(is, not, a)
- List(not, a, test)
- List(This, is, not)

Correct!

Question 23

1 / 1 pts

Consider the following Scala list:

```
val list = "This" :: "is" :: "not" :: "a" :: "test" :: Nil
```

What would list.filter(s => s.length < 4) return?

- List(This, test)
- List(is, not, a)
- List(This, is, not, a, test)
- List()

Correct!

Question 24

1 / 1 pts

Consider the following Scala list:

```
val list = "This" :: "is" :: "not" :: "a" :: "test" :: Nil
```

What would list.map(s => s.length + 1) return?

- List((This -> 1), (is -> 1), (not -> 1), (a -> 1), (test -> 1))
- List(5, 3, 4, 2, 5)
- List(4This, 2This, 3This, 1This, 4This)
- List(This1, is1, not1, a1, test1)

Correct!

Question 25

1 / 1 pts

[Project 2]

Consider the following representation of a circular array pipe.

contents = [A, B, C, D] and first = 2 and last = 0 and length = 3

Which of the following pipe abstractions correspond to this representation?

Correct!

- (●) [C, D, A]
- ( ) [A, B, C]
- ( ) [A, B]
- ( ) [B, C, D]

Quiz Score: **24** out of 25