Started on	Sunday, 9 May 2021, 15:09
State	Finished
Completed on	Sunday, 9 May 2021, 16:41
Time taken	1 hour 31 mins
Grade	**50.00** out of 150.00 (**33**%)

Question 1

Incorrect Mark 0.00 out of 10.00

Consider an abstract data type `Bool`.
The type has the following operations:

```
true : Bool
false : Bool

and : Bool × Bool → Bool
or : Bool × Bool → Bool
not : Bool → Bool
```

where the first two operations construct the two values of the type,
and last three operations have the usual meanings of logical *and*, logical *or*, and logical *not* on those values.

The following are possible ways that `Bool` might be implemented and still be able to satisfy the specs of the operations, except one.
Which one is **not** the correct way?

Select one:

a. As a `long` value in which all possible values mean `true`.

b. As a single bit, where 1 means `true` and 0 means `false`.

c. As an `int` value where 2 means `true` and 5 means `false`.

d. As a reference to a `String` object where `"false"` to mean `true` and `"true"` to mean `false`

Your answer is incorrect.
The correct answer is: As a `long` value in which all possible values mean `true`.

Question 2

Incorrect Mark 0.00 out of 10.00

The method below is an operation on an abstract data type from the Java library.

It is followed by the link of its documentation.

Read it, and classify the operation :

Integer.valueOf()

https://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html#valueOf-java.lang.String-

Select one:

a. creator

b. producer

c. mutator

d. observer

Your answer is incorrect.

The correct answer is: creator

Question 3

Correct Mark 10.00 out of 10.00

The method below is an c

It is followed by the link of its documentation.

Read it, and classify the operation :

BigInteger.mod()

https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html#mod-java.math.BigInteger-

Select one:

a. creator

b. producer

c. mutator

d. observer

Your answer is correct.

The correct answer is: producer

Question 4

Incorrect Mark 0.00 out of 10.00

The method below is an operation on an abstract data type from the Java library.

It is followed by the link of its documentation.

Read it, and classify the operation :

List.addAll()

https://docs.oracle.com/javase/8/docs/api/java/util/List.html#addAll-java.util.Collection-

Select one:

- a. creator
- b. producer
- c. mutator
- d. observer

Your answer is incorrect.

The correct answer is: mutator

Question 5

Correct Mark 10.00 out of 10.00

The method below is an operation on an abstract data type from the Java library.

It is followed by the link of its documentation.

Read it, and classify the operation :

Collections.unmodifiableList()

https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html#unmodifiableList-java.util.List-

Select one:

- a. creator
- b. producer
- c. mutator
- d. observer

Your answer is correct.

The correct answer is: producer

Question 6

Correct Mark 10.00 out of 10.00

The method below is an operation on an abstract data type from the Java library.

It is followed by the link of its documentation.

Read it, and classify the operation :

String.toUpperCase()

https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#toUpperCase--

Select one:

a. creator

b. producer

c. mutator

d. observer

Your answer is correct.

The correct answer is: producer

Question 7

Correct Mark 10.00 out of 10.00

The method below is an operation on an abstract data type from the Java library.

It is followed by the link of its documentation.

Read it, and classify the operation :

Set.contains()

https://docs.oracle.com/javase/8/docs/api/java/util/Set.html#contains-java.lang.Object-

Select one:

a. creator

b. producer

c. mutator

d. observer

Your answer is correct.

The correct answer is: observer

Question 8

Incorrect Mark 0.00 out of 10.00

The method below is an operation on an abstract data type from the Java library.

It is followed by the link of its documentation.

Read it, and classify the operation :

BufferedReader.readLine()

https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html#readLine--

Select one:

a. creator

b. producer

b. producer

c. mutator

d. observer

Your answer is incorrect.

The correct answer is: mutator

Question 9

Incorrect | Mark 0.00 out of 10.00

Consider the following abstract data type.

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family, sorted from oldest to youngest, with no duplicates.
    public List<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return people;
    }
}
```

Here is a client of this abstract data type:

```
void client1(Family f) {
    // get youngest person in the family
    Person baby = f.people.get(f.people.size() - 1);
    ...
}
```

Assume all this code works correctly (both `Family` and `client1`) and passes all its tests.

Now `Family`'s representation is changed from a `List` to `Set`, as shown:

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family
    public Set<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return new ArrayList<>(people);
    }
}
```

Assume that `Family` compiles correctly after the change.

Which of the following statements are true about `client1` after `Family` is changed?

Select one:

a. `client1` is independent of `Family`'s representation, so it keeps working correctly.

b. `client1` depends on `Family`'s representation, and the dependency would be caught as a static error.

c. `client1` depends on `Family`'s representation, and the dependency would be caught as a dynamic error.

d. `client1` depends on `Family`'s representation, and the dependency would not be caught but would produce a wrong answer at runtime.

e. `client1` depends on `Family`'s representation, and the dependency would not be caught but would (luckily) still produce the same answ

Your answer is incorrect.

The correct answer is: `client1` depends on `Family`'s representation, and the dependency would be caught as a static error.

Question 10

Incorrect | Mark 0.00 out of 10.00

Consider the following abstract data type.

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family, sorted from oldest to youngest, with no duplicates.
    public List<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return people;
    }
}
```

Here is a client of this abstract data type:

```
void client2(Family f) {
    // get size of the family
    int familySize = f.people.size();
    ...
}
```

Assume all this code works correctly (both `Family` and `client2`) and passes all its tests.

Now `Family`'s representation is changed from a `List` to `Set`, as shown:

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family
    public Set<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return new ArrayList<>(people);
    }
}
```

Assume that `Family` compiles correctly after the change.

Which of the following statements are true about `client2` after `Family` is changed?

Select one:

a. `client2` is independent of `Family`'s representation, so it keeps working correctly.

b. `client2` depends on `Family`'s representation, and the dependency would be caught as a static error.

c. `client2` depends on `Family`'s representation, and the dependency would be caught as a dynamic error.

d. `client2` depends on `Family`'s representation, and the dependency would not be caught but would produce a wrong answer at runtim

e. `client2` depends on `Family`'s representation, and the dependency would not be caught but would (luckily) still produce the same answ

Your answer is incorrect.

The correct answer is: `client2` depends on `Family`'s representation, and the dependency would not be caught but would (luckily) still produce the same answer.

Question 11

Incorrect | Mark 0.00 out of 10.00

Consider the following abstract data type.

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family, sorted from oldest to youngest, with no duplicates.
    public List<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return people;
    }
}
```

Here is a client of this abstract data type:

```
void client3(Family f) {
    // get any person in the family
    Person anybody = f.getMembers().get(0);
    ...
}
```

Assume all this code works correctly (both `Family` and `client3`) and passes all its tests.

Now `Family`'s representation is changed from a `List` to `Set`, as shown:

```
/**
 * Represents a family that lives in a household together.
 * A family always has at least one person in it.
 * Families are mutable.
 */
class Family {
    // the people in the family
    public Set<Person> people;

    /**
     * @return a list containing all the members of the family, with no duplicates.
     */
    public List<Person> getMembers() {
        return new ArrayList<>(people);
    }
}
```

Assume that `Family` compiles correctly after the change.

Which of the following statements are true about `client3` after `Family` is changed?

Select one:

a. `client3` is independent of `Family`'s representation, so it keeps working correctly.

b. `client3` depends on `Family`'s representation, and the dependency would be caught as a static error.

c. `client3` depends on `Family`'s representation, and the dependency would be caught as a dynamic error.

d. `client3` depends on `Family`'s representation, and the dependency would not be caught but would produce a wrong answer at runtim

e. `client3` depends on `Family`'s representation, and the dependency would not be caught but would (luckily) still produce the same answ

Your answer is incorrect.

The correct answer is: `client3` is independent of `Family`'s representation, so it keeps working correctly.

Question 12

Correct Mark 10.00 out of 10.00

```
1  /**
2   * Represents a family that lives in a household together.
3   * A family always has at least one person in it.
4   * Families are mutable.
5   */
6  public class Family {
7      // the people in the family, sorted from oldest to youngest, with no duplicates.
8      private List<Person> people;
9
10     /**
11      * @return a list containing all the members of the family, with no duplicates.
12      */
13     public List<Person> getMembers() {
14         return people;
15     }
16 }
```

Which line is part of the representations?

Select one:

- a. lines 1-5
- b. line 6
- c. line 8
- d. lines 10-12
- e. line 13
- f. line 14

Your answer is correct.

The correct answer is: line 8

Question 13

Incorrect Mark 0.00 out of 10.00

```
1  /**
2   * Represents a family that lives in a household together.
3   * A family always has at least one person in it.
4   * Families are mutable.
5   */
6  public class Family {
7      // the people in the family, sorted from oldest to youngest, with no duplicates.
8      private List<Person> people;
9
10     /**
11      * @return a list containing all the members of the family, with no duplicates.
12      */
13     public List<Person> getMembers() {
14         return people;
15     }
16 }
```

Which line is part of the implementations?

Select one:

- a. lines 1-5

b. line 6

c. line 8

d. lines 10-12

e. line 13

f. line 14

Your answer is incorrect.

The correct answer is: line 14

Question 14

Incorrect Mark 0.00 out of 10.00

Choose the correct statement.

Select one:

a. If you are a subclass of an interface, you have to override all of its method signatures.

b. If you override a method, you have to annotate the method with @Override

c. Method overloading is when you have multiple methods with the same signature, but different names.

d. An object o is instantiated with static type S and dynamic type D.
D is a subclass of S, and D overloads method m() of S.
At runtime, o.m() will call method m() that belongs to D.

Your answer is incorrect.

The correct answer is: If you are a subclass of an interface, you have to override all of its method signatures.

Question 15

Incorrect Mark 0.00 out of 10.00

Which statement is **incorrect** about default method ?

Select one:

a. Default method must be overridden by the subclass of the interface.

b. Default method is implemented in an interface.

c. Default method is inherited by the subclass of the interface.

d. Default method that is overridden by a subclass of the interface will be run because of the dynamic method selection.

Your answer is incorrect.

The correct answer is: Default method must be overridden by the subclass of the interface.

Finish review

Lab 10 Recording

Jump to...

Lab Exercise 10.1 ARDeque DE