

Inheritance

Preliminary

The following questions are not going to be discussed. Answers is given without explanation. You are encouraged to discuss them on Ed.

1. Consider the following classes.

Time

```
/**
 * The class encapsulates a time measurement in units of seconds and milliseconds.
 */
class Time {
  public int second;
  public int millisecond;

  public Time(int second, int millisecond) {
    this.second = second;
    this.millisecond = millisecond;
  }
}
```

Interval

```
/**
 * The class encapsulates a time interval consisting of starting time and ending
time.
 */
class Interval {
  private Time begin;
  private Time end;

  public Interval(Time begin, Time end) {
    this.begin = begin;
    this.end = end;
  }

  /**
   * Returns the duration between begin and end in millisecond.
   */
  public int durationInMs() {
    return (end.second * 1000 + end.millisecond)
```

```
      - (begin.second * 1000 + begin.millisecond);
  }
}
```

Jiffy

```
/**
 * The class encapsulates a time interval with a fixed duration of 10 milliseconds.
 */
class Jiffy extends Interval {
  public Jiffy() {
    super(new Time(0, 0), new Time(0, 10));
  }

  @Override
  /**
   * Returns 10.
   */
  public int durationInMs() {
    return 10;
  }
}
```

(a) What is the specification of `Interval::durationInMs()`?

(b) What is the specification of `Jiffy::durationInMs()`?

(c) "The code above violates information hiding principle." Is the given statement true or false? Explain your answer.

(d) "The code above violates "Tell, Don't Ask" principle." Is the given statement true or false? Explain your answer.

(e) "The code above violates Liskov substitution principle[1]." Is the given statement true or false? Explain your answer.

2. Consider the following interfaces `CanDoA` and `CanDoB`.

CanDoA

```
public interface CanDoA {
  void doA();
}
```

CanDoB

```
public interface CanDoB {
  void doB();
}
```

(a) Implement the class `C` that implements both interfaces.

(b) Consider the following code snippet.

Code Snippet

```
C c = new C();
CanDoA a = c;
CanDoB b = c;
```

Which of the following statements are allowed?

(i) `a.doA();`

(ii) `a.doB();`

(iii) `b.doA();`

(iv) `b.doB();`

Discussion

The following questions are problem sets to be discussed during recitation. You are expected to attempt them before coming to the recitation. You may be randomly called to present your answer during recitation. You are encouraged to discuss them on Ed.

1. Consider the following definition of `Rectangle` class.

Rectangle

```
/**
 * The class encapsulates a rectangle with a fixed width and height.
 */
public class Rectangle {
  private double width;
  private double height;

  public Rectangle(double width, double height) {
    this.width = width;
    this.height = height;
  }

  /**
   * Returns the area of a rectangle as width * height.
   */
  public double getArea() {
    return this.width * this.height;
  }

  @Override
  public String toString() {
    return "Width: " + this.width + " Height: " + this.height;
  }
}
```

The method `Rectangle::area` is expected to return the product of its width and height.

Sample Run

```
jshell> new Rectangle(5, 8);
$.. ==> Width: 5.0 Height: 8.0
jshell> new Rectangle(5, 8).getArea();
$.. ==> 40.0
```

Consider the following specification of a **square**.

> *A **square** has four sides of the same length. All four sides are of fixed length. In other words, its width is equal to its height. Given a square with a length L, its area is $L \times L$. We can get the area using the method `getArea()`.*

Sample Run

```
jshell> new Square(5);
$.. ==> Width: 5.0 Height: 5.0
jshell> new Square(5).getArea();
$.. ==> 25.0
```

(a) Design the class `Square` using class diagram. Ensure that your design follows a good object-oriented programming principles.

(b) Implement the class `Square` satisfying good object-oriented programming principles.

(c) Explain why your implementation satisfying good object-oriented programming principles.

(d) Consider an extension of the specification such that both `Rectangle` and `Square` can have updatable sides. The sides of a `Square` must still satisfy the condition that it has equal length. Further, we added two methods to `Rectangle` as follows.

setH

```
/**
 * Modifies the height to the input.
 */
public void setH(double val) {
  this.height = val;
}
```

setW

```
/**
 * Modifies the width to the input.
 */
public void setW(double val) {
  this.width = val;
}
```

We now require `area` to return `width * height` such that `width` is from the latest width as updated by `setW` and `height` is from the latest height as updated by `setH`. Redesign and re-implement `Square` satisfying good object-oriented programming principles.

2. Consider the following collection of classes.
   - `Shape` must have the method `double getArea()`.
   - `Surface` must have the method `double perimeter()`.

   A `Circle` must define both `double getArea()` and `double perimeter()`.

   (a) Suppose someone proposes to code both `Shape` and `Surface` as abstract classes. Would this work? Explain.
   Hint: Consider how C++[2] and Python[3] solved this.

   (b) Suppose someone proposes to code both `Shape` and `Surface` as interfaces. Would this work? Explain.

   (c) Suppose someone proposes to code both `Shape` and `Surface` as interface and define another interface `Blob` that implements `Shape` and `Surface`. Would this work? Explain.

Exercise

The following questions are to be discussed on Ed as no answer will be given. ***Use only concepts that you have learnt****. You are expected to attempt them before the next lecture as the concepts may be used and/or mentioned in the lecture. The concept may also be tested for assessment.*

1. Let us create a `Box`. The `Box` can only store one value in a field named `val`. It should implement the following simple methods.

   (i) A constructor that accepts a value and assign it to `val`.

   (ii) An accessor `getValue()` that returns the value.

   Eventually, our `Box` should be able to store any type.

   (a) Create a class `BoxOfString` that stores a `String`.

   (b) Create a class `BoxOfInteger` that stores a `Integer`.

   (c) Create a class `BoxOfInt` that stores a `int`. How different is your implementation from the previous code.

   (d) Generalize the code to create a class `Box` that can store an object of any reference type.

   (e) Is your implementation of `Box` type safe?

Is it possible to store a `String` and retrieve a `String` back? However, you may need to add other instructions (e.g., typecast). Can it compile? Can it run successfully?

Sample Run

```
Box box = new Box("CS2030S");
String val = box.getValue();
```

Is it possible to store a `String` and retrieve a `Integer` back similar to below? You need to use similar technique to above. Can it compile? Can it run successfully?

Sample Run

```
Box box = new Box("CS2030S");
Integer val = box.getValue();
```

Bibliography

[1] B. H. Liskov and J. M. Wing, "A behavioral notion of subtyping," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 6, pp. 1811–1841, Nov. 1994.

[2] I. CPP, "C++: Dreaded Diamond." [Online]. Available: https://isocpp.org/wiki/faq/multiple-inheritance/#mi-diamond

[3] Python.org, "Python 2.3: Method Resolution Order." [Online]. Available: https://www.python.org/download/releases/2.3/mro/

[4] Oracle, "Default Method." [Online]. Available: https://docs.oracle.com/javase/tutorial/java/IandI/defaultmethods.html

[5] Oracle, "Inheritance." [Online]. Available: https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html