[M2] Individual Readiness Assurance Test (iRAT) Results for Christine En-Tse Cheng

Score for this attempt: **4** out of 4
Submitted Oct 2 at 6:24p.m.
This attempt took 9 minutes.

Quiz Submissions
Attempt 1: 4
Christine En-Tse Cheng has 1 attempt left
← Back to Quiz

Correct answer

Question 1

1 / 1 pts

Consider a program where some of the classes are required to be able to split a file. Other classes are required to be able to split a list. There is one class that is required to be able to split both files and lists.

If these classes all implement the following interface, **which SOLID principle is most directly violated?**

```
public interface Splittable {
  void prepareFileForSplit();
  boolean fileSplit(float percentageWhereToSplit);
  void prepareListForSplit();
  boolean listSplit(int indexWhereToSplit);
}
```

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- ✓ Interface Segregation Principle
- Dependency Inversion Principle

Correct answer

Question 2

1 / 1 pts

Consider a program that contains a class called `PetSoundGenerator`. It contains private methods called `findSound`, `soundToString`, and `verifySound`. Its only public method is:

```
public String makeSomeNoise(Dog d) {
  String res = findSound(d);
  if(verifySound(res)) {
    return soundToString(res);
  } else {
    return "No sound";
  }
}
```

In order to allow the program to work with other animals besides dogs, we introduce a parent class for `Dog` called `Animal`. Then we replace the `Dog` parameter in the `makeSomeNoise` method with `Animal`.

While this change may align with more than one SOLID principle, if our goal is primarily to make it easier to add new animal classes in the future, **which SOLID principle does this best exemplify?**

- Single Responsibility Principle
- ✓ Open/Closed Principle
- Liskov Substitution Principle
- Interface Substitution Principle
- Dependency Inversion Principle

Correct answer

Question 3

1 / 1 pts

In the previous question, we changed the `makeSomeNoise` method to depend on class `Animal` instead of its subclass `Dog`. While this change may align with more than one SOLID principle, **which principle** specifically encourages us to depend on abstractions like `Animal` rather than more concrete classes like `Dog`?

- Single Responsibility Principle
- Open/Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- ✓ Dependency Inversion Principle

Correct answer

Question 4

1 / 1 pts

Consider a program that allows a user to buy online tickets for different events with the possibility of returning a ticket for a full refund.

Originally, the program included the following class called Ticket:

```
public class Ticket {

  boolean isForSale;

  // Constructor code is omitted

  public void updateTicket(boolean isForSale) {
    this.isForSale = isForSale;
  }
}
```

Then we introduce the following subclass:

```
public class MovieTicket extends Ticket {

  boolean isForSale;

  // Constructor code is omitted

  @Override
  public void updateTicket(boolean isForSale) {
    this.isForSale = true;
  }
}
```

This is a violation of which SOLID principle? Choose the one most specifically describing this particular situation.

- Single Responsibility Principle
- Open/Closed Principle
- ✓ Liskov Substitution Princple
- Interface Segregation Principle
- Dependency Inversion Principle

Quiz Score: **4** out of 4