Inheritance DOs and DON'Ts

CSCI 2300

Inheritance Pros and Cons

Advantages

Disadvantages

```
public class Topping
{
      private String topping;
      public Topping(String t)
      {
          this.topping = t;
      }
      String get()
      {
          return this.topping;
      }
}
```

```
public class Pizza
{
      public Topping[] get Toppings()
      {
          return null;
      }
}
```

Does PepperoniPizza class work?

```
public class PepperoniPizza extends Pizza
{
      Topping pepperoni;
      public PepperoniPizza()
      {
          pepperoni = new Topping("pepperoni");
      }
      public Topping[] getToppings()
      {
          Topping [] topping = {pepperoni};
          return topping;
      }
}
```

A. Yes
B. No – Pizza does not have a constructor
C. No – Topping does not have a no-argument constructor
D. No – you cannot return an array from a method


```
public class Topping
{
      private String topping;
      public Topping(String t)
      {
          this.topping = t;
      }
      String get()
      {
          return this.topping;
      }
}
```

```
public class Pizza
{
      public Topping[] get Toppings()
      {
          return null;
      }
}
```

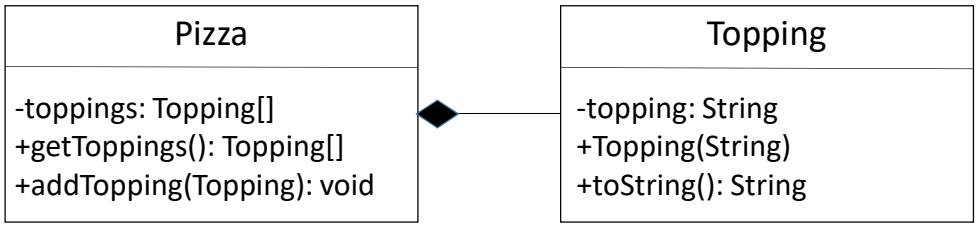
```
public class PepperoniPizza extends Pizza
{
      Topping pepperoni;
      public PepperoniPizza()
      {
          pepperoni = new Topping("pepperoni");
      }
      public Topping[] getToppings()
      {
          Topping [] topping = {pepperoni};
          return topping;
      }
}
```


Is this a good design?

Don't abuse inheritance!

A subclass must have some attributes or behaviors that don't belong in parent class

Redesigned Pizza



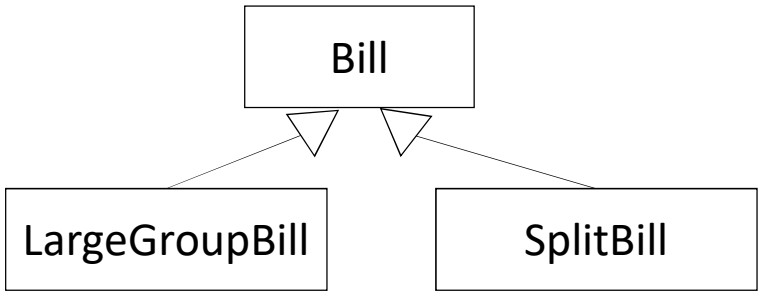
Liskov Substitution Principle (LSP)

- Subtypes can be substituted for their base types
- Subclass IS-SUBSTITUTABLE-FOR base class

Violation of LSP

- Class “Rectangle”
- Class “Square” extends “Rectangle”
- “Square” enforces that height and width are equal
- Suppose the following code:

```
public double area(Rectangle r)
{
    r.setHeight(10);
    r.setWidth(5);
    return r.getArea();
}
```




```
void process(Bill b)
{
    if (b instanceof SplitBill)
    {
    }
}
```

`instanceof` is a Java operator that evaluates to true if the given object is an instance of the given type

```
public double calculate (Bill bill)
{
   if (bill instanceof LargeGroupBill)
   {
      // add up the bill items and add 15% gratuity
   }
   else
   {
      // add up the bill items }
   }
}
```

Does this code violate LSP?

A. This code violates LSP: we are checking the sub-type of Bill to determine the logic
B. This code violates LSP: the calculation should be done in the Bill class
C. This code violates LSP because LargeGroupBill is not defined
D. This code does not violate LSP

```
public class Vehicle{
    public void drive(int miles){
        if (miles > 0 && miles < 300){…}
    }
}
```

```
public class Scooter extends Vehicle{
     public void drive(int miles){
          if (miles > 0 && miles < 50){
              super.drive(miles);
          }
     }
 }
```

Does this code violate LSP?

A. This code does not violate LSP.
B. This code violates LSP because it restricts the behavior of Vehicle.
C. This code violates LSP because the drive() method of Scooter calls parent's drive() method.

```
class ToyCar extends Vehicle{
    public void drive(int miles) {
      // Show flashy lights, make random sounds
    }
    public void fillUpWithFuel() {
       // silly lights and noises
    public int fuelRemaining { return 0;}
}
```

Does this code violate LSP?

A. This code does not violate LSP
B. This code violates LSP because it completely changes the behavior of drive() and fillUpWithFuel()
C. This code violates LSP because fuelRemaining() returns 0
D. B and C.

Don't abuse inheritance!

An object of a subclass must be SUBSTITUTABLE for an object of a parent class

Lab 5

- Create lab5 in your csci2300 git repos
- Redesign classes from the Pizza example to not abuse inheritance
- Create a Driver class that creates and prints a Pizza object
- Commit and push your code