CSE 331: Subtyping

Consider this code:

```
public void foo(String s) { ... }
...
foo(17);  // type error! foo expects String but we pass int
```

So, is it always true that we must pass exactly the expected type? No!

```
public class Product {
  private String name;
  private double price;
  public Product(String name, double price) { ... }
  public double getPrice() { return price; }
  public String toString() { return name + ": $" + price; }
}
public class SaleProduct extends Product {
  private double discountRate;
  public SaleProduct(String name, double price, double discountRate) { ... }
  public double getPrice() { return price * discountRate; }
}
public static void main(String[] args) {
  List<Product> l = new ArrayList<>();
  l.add(new Product("skittles", 1.0));
  l.add(new SaleProduct("m&ms", 1.0, 0.9));
  for (Product p : l) {
    System.out.println(p);
  }
}
```

That code passes a `SaleProduct` where a `Product` is expected.
What is the alternative? _____

There are several related but distinct concepts:
- Inheritance: Copying fields and methods from superclass to subclass.
- Behavioral subtype: a B is substitutable wherever an A is expected.
- Java subtype: `B extends A` or `B implements A` (or transitively).
- Java subclassing: `B extends A`

```
public class Rectangle {
  private int width;
  private int height;
  public Rectangle(int w, int h) { ... }
  public int getWidth() { ... }
  public int getHeight() { ... }
  public void setWidth(int w) { ... }
  public void setHeight(int h) { ... }
}

public class Square extends Rectangle {
  public void setWidth(int w) {
    ???
  }
}
```

Type B is a **behavioral subtype** of type A if every object satisfying B's spec also satisfies A's spec.

Liskov substitution principle: when B is a behavioral subtype of A, one can use a B wherever an A is expected.