Class Design

Using this() and super()

```java
// special method this() is used to call another constructor in a constructor


public class Dog {
  private String name;
  private int age;
  public Dog(String name, int age) {
    this.name = name;
    this.age = age;
  }
  public Dog() {
    this("Chip", 1);
    System.out.println("Woof!");
  }
}
```

```java
// main method
public static void main(String[] args) {
  Dog dog = new Dog();
  System.out.println("Name: " + dog.name + ", " + "Age: " + dog.age);
}
```

```
Woof!
Name: Chip, Age: 1
```

```java
public Dog() {
  this("Chip", 1);
  System.out.println("Woof!");
}
```

Rules for using this()

1. `this()` can only be called **in the first line** in the constructor
2. `this()` can be called **only once**
3. you must be careful not to create a "cycle"
   - constructors which call each other *ad infinitum*

```
// example #1
public Dog() {
  System.out.println("Woof!");
  this("Chip", 1);
}
```

does not compile

```
// example #2
public Dog() {
  this();
  System.out.println("Woof!");
}
```

cycle (infinite loop)

```
// example #3
public Dog() {
  this("Chip", 1);
  System.out.println("Woof!");
}

public Dog(String name, int age) {
  this();
  this.name = name;
  this.age = age;
}
```

cycle (infinite loop)

```
// special method super() is used to call a constructor of a superclass
// in a constructor of the subclass

class Mammal {
  public int age;
  public Mammal(int age) { this.age = age; }
}
public class Dog extends Mammal {
  private String name;
  public Dog(String name, int age) {
    super(age);
    this.name = name;
    System.out.println("Woof!")
  }
}
```

```
// if there is no this() or super() in the first line
// then the compiler will insert super() automatically
```

!!!

```
class Mammal {
  public int age;
}
public class Dog extends Mammal {
  private String name;
  public Dog() {
    System.out.println("Woof!")
  }
}
```

the compiler inserts `super();`

```
// be careful if the superclass doesn't have the no-argument constructor


class Mammal {
  public int age;
  public Mammal(int age) { this.age = age; }
}
public class Dog extends Mammal {
  private String name;
  public Dog() {
    System.out.println("Woof!")
  }
}
```

we have provided a constructor (with arguments),

therefore the compiler will not auto-generate no-arg constructor

the compiler inserts `super();`

there is no `Mammal()` constructor => does not compile

Rules for using super()

1. If there is no explicit `this()` or `super()` in the first line of the constructor
   - the compiler will insert `super()` at the beginning of every constructor
2. Can be called only once
3. Must be called in the first line of the constructor