CSCI 132:
Basic Data Structures & Algorithms

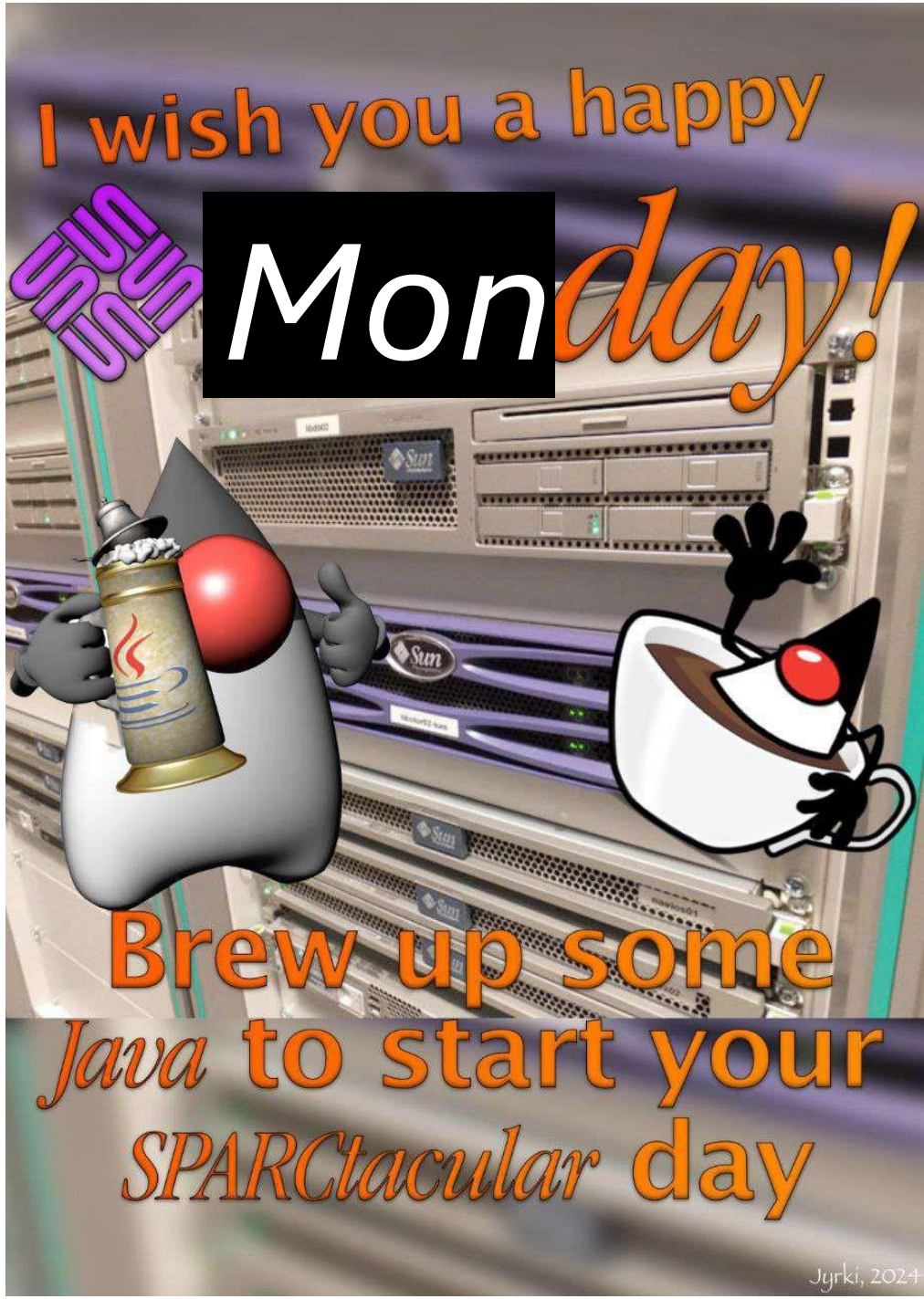
Inheritance

Reese Pearsall & Iliana Castillon
Fall 24

MONTANA
STATE UNIVERSITY

Announcements

- Lab 3 Posted
- Comment ~75% of methods in Program 1
- Program 1
  - Start small!
  - Break it down into pieces
  - Think about what could be grouped together
    - Maybe add those items to the same class!



Inheritance

- **Inheritance** is a mechanism in Java that allows for a class to acquire instance fields and methods from another class
- In Java, we use the **`extends`** keyword to indicate that a class is inheriting from another

```
public class Vegetable extends Food{
}
```

```java
public class Vegetable extends Food{

    private String growthSeason;
    private boolean isLeafy;

    Vegetable(String name, int calores, double price, String growthSeason, boolean isLeafy){
        super(name, calores, price);
        this.growthSeason = growthSeason;
        this.isLeafy = isLeafy;
    }

    public String getGrowthSeason() {
        return this.growthSeason;
    }

    public boolean getIsLeafy() {
        return this.isLeafy;
    }

    public void testMethod() {
        System.out.println("I am inside the Vegetable class");
    }
}
```

Vegetable.java

```java
public abstract class Food {

    private String name;
    private double calores;
    private double price;

    public Food(String name, int calories, double price) {
        this.name = name;
        this.calores = calories;
        this.price = price;
    }

    public String getName() {
        return this.name;
    }

    public double getCalories() {
        return this.calores;
    }

    public double getPrice() {
        return this.price;
    }

    public void testMethod() {
        System.out.println("I am inside the Food class");
    }

}
```

Food.java

The `Vegetable` class inherits from the `Food` class

```java
public class Vegetable extends Food{

    private String growthSeason;
    private boolean isLeafy;

    Vegetable(String name, int calores, double price, String growthSeason, boolean isLeafy){
        super(name, calores, price);
        this.growthSeason = growthSeason;
        this.isLeafy = isLeafy;
    }

    public String getGrowthSeason() {
        return this.growthSeason;
    }

    public boolean getIsLeafy() {
        return this.isLeafy;
    }

    public void testMethod() {
        System.out.println("I am inside the Vegetable class");
    }
}
```

Vegetable.java

```java
public abstract class Food {

    private String name;
    private double calores;
    private double price;

    public Food(String name, int calories, double price) {
        this.name = name;
        this.calores = calories;
        this.price = price;
    }

    public String getName() {
        return this.name;
    }

    public double getCalories() {
        return this.calores;
    }

    public double getPrice() {
        return this.price;
    }

    public void testMethod() {
        System.out.println("I am inside the Food class");
    }

}
```

Food.java

```java
Vegetable spinach = new Vegetable("Spinach",7,3,"Fall",true);
System.out.println(spinach.getName());
```

getName() is not defined in the Vegetable class, but because the Vegetable class *inherits* from the Food class, the apple object has access to the getName() method

```java
public class Vegetable extends Food{

    private String growthSeason;
    private boolean isLeafy;

    Vegetable(String name, int calores, double price, String growthSeason, boolean isLeafy){
        super(name, calores, price);
        this.growthSeason = growthSeason;
        this.isLeafy = isLeafy;
    }

    public String getGrowthSeason() {
        return this.growthSeason;
    }

    public boolean getIsLeafy() {
        return this.isLeafy;
    }

    public void testMethod() {
        System.out.println("I am inside the Vegetable class");
    }
}
```

Vegetable.java

```java
public abstract class Food {

    private String name;
    private double calores;
    private double price;

    public Food(String name, int calories, double price) {
        this.name = name;
        this.calores = calories;
        this.price = price;
    }

    public String getName() {
        return this.name;
    }

    public double getCalories() {
        return this.calores;
    }

    public double getPrice() {
        return this.price;
    }

    public void testMethod() {
        System.out.println("I am inside the Food class");
    }
}
```

Inherited!

Food.java

```java
Vegetable spinach = new Vegetable("Spinach",7,3,"Fall",true);
System.out.println(spinach.getName());
```

getName() is not defined in the Vegetable class, but because the Vegetable class *inherits* from the Food class, the apple object has access to the getName() method

```java
public class Vegetable extends Food{

    private String growthSeason;
    private boolean isLeafy;

    Vegetable(String name, int calores, double price, String growthSeason, boolean isLeafy){
        super(name, calores, price);
        this.growthSeason = growthSeason;
        this.isLeafy = isLeafy;
    }

    public String getGrowthSeason() {
        return this.growthSeason;
    }

    public boolean getIsLeafy() {
        return this.isLeafy;
    }

    public void testMethod() {
        System.out.println("I am inside the Vegetable class");
    }
}
```

Vegetable.java

```java
public abstract class Food {

    private String name;
    private double calores;
    private double price;

    public Food(String name, int calories, double price) {
        this.name = name;
        this.calores = calories;
        this.price = price;
    }

    public String getName() {
        return this.name;
    }

    public double getCalories() {
        return this.calores;
    }

    public double getPrice() {
        return this.price;
    }

    public void testMethod() {
        System.out.println("I am inside the Food class");
    }

}
```

Not inherited! (but getter methods are)

Food.java

private instance fields and methods are not inherited

```java
public class Vegetable extends Food{

    private String growthSeason;
    private boolean isLeafy;

    Vegetable(String name, int calores, double price, String growthSeason, boolean isLeafy){
        super(name, calores, price);
        this.growthSeason = growthSeason;
        this.isLeafy = isLeafy;
    }

    public String getGrowthSeason() {
        return this.growthSeason;
    }

    public boolean getIsLeafy() {
        return this.isLeafy;
    }

    public void testMethod() {
        System.out.println("I am inside the Vegetable class");
    }
}
```

Vegetable.java

```java
public abstract class Food {

    protected String name;
    protected double calores;
    protected double price;

    public Food(String name, int calories, double price) {
        this.name = name;
        this.calores = calories;
        this.price = price;
    }

    public String getName() {
        return this.name;
    }

    public double getCalories() {
        return this.calores;
    }

    public double getPrice() {
        return this.price;
    }

    public void testMethod() {
        System.out.println("I am inside the Food class");
    }

}
```

Food.java

Now this instance fields will be inherited 😎

private instance fields and methods are not inherited

We can make instance fields **protected**, which means they are still private to other classes, but now they can be inherited

```java
public class Vegetable extends Food{

    private String growthSeason;
    private boolean isLeafy;

    Vegetable(String name, int calores, double price, String growthSeason, boolean isLeafy){
        super(name, calores, price);
        this.growthSeason = growthSeason;
        this.isLeafy = isLeafy;
    }

    public String getGrowthSeason() {
        return this.growthSeason;
    }

    public boolean getIsLeafy() {
        return this.isLeafy;
    }

    public void testMethod() {
        System.out.println("I am inside the Vegetable class");
    }
}
```

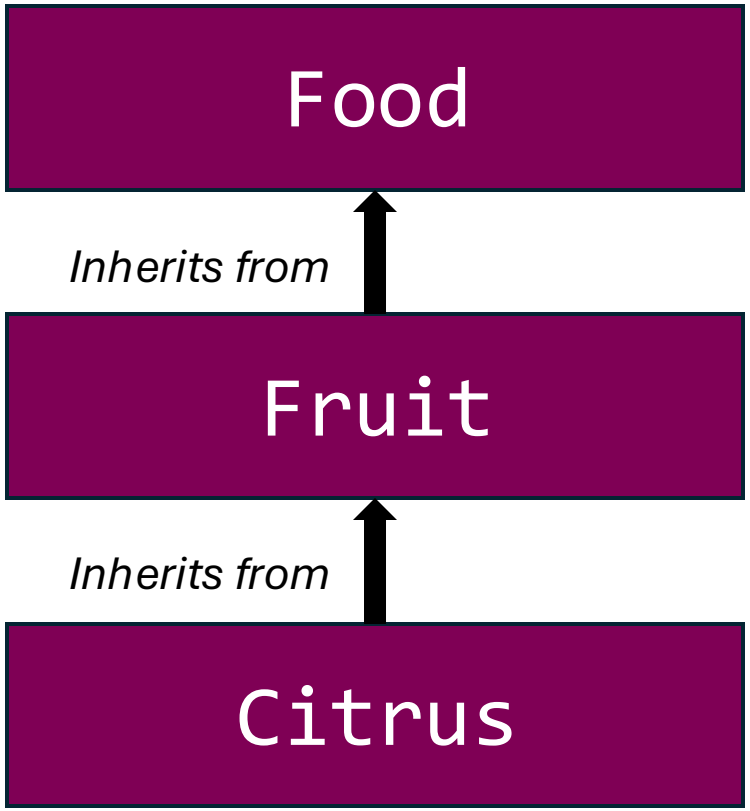
Vegetable.java

```java
public abstract class Food {

    protected String name;
    protected double calores;
    protected double price;

    public Food(String name, int calories, double price) {
        this.name = name;
        this.calores = calories;
        this.price = price;
    }

    public String getName() {
        return this.name;
    }

    public double getCalories() {
        return this.calores;
    }

    public double getPrice() {
        return this.price;
    }

    public void testMethod() {
        System.out.println("I am inside the Food class");
    }

}
```

Food.java

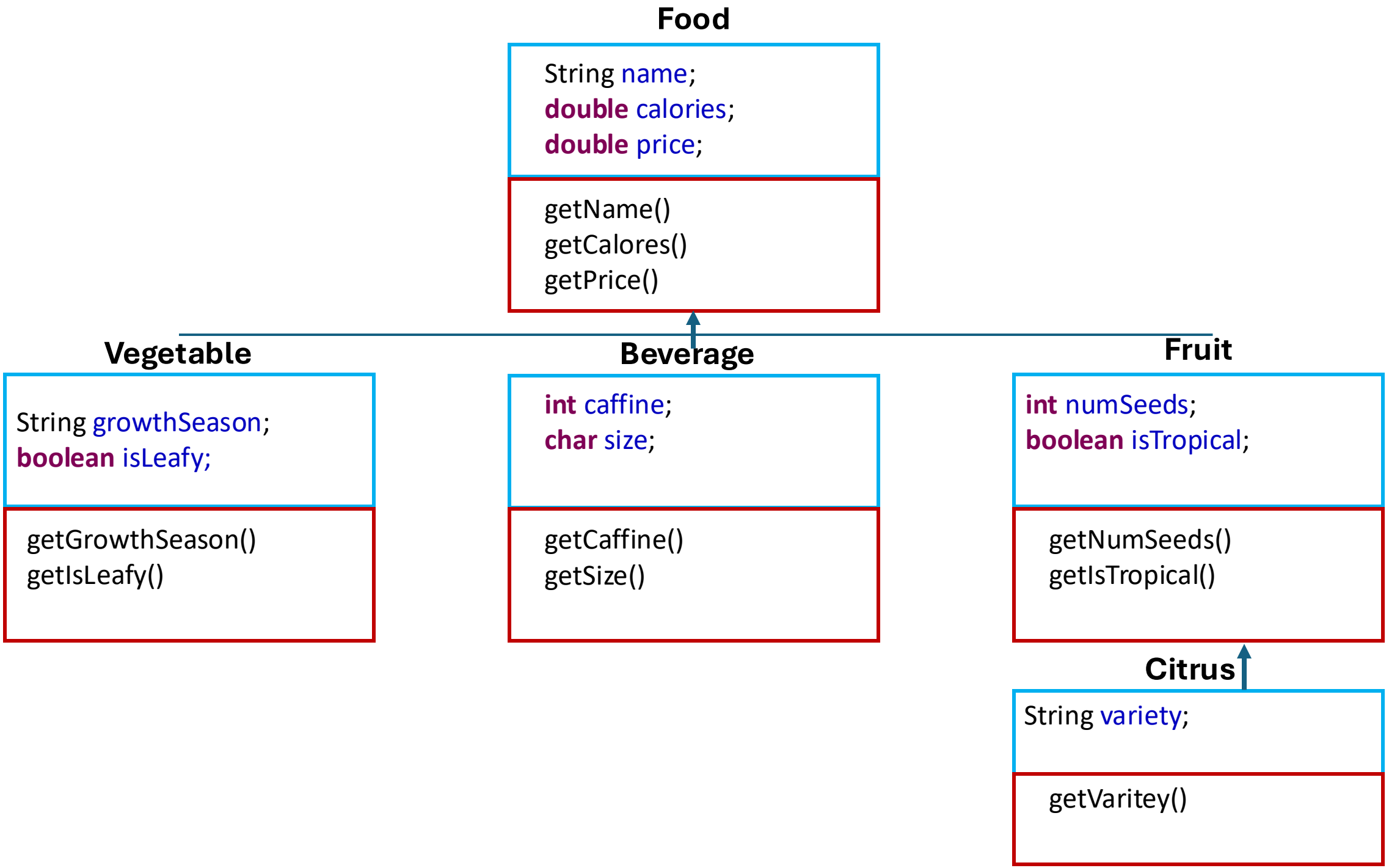
The **super** keyword is used to reference the parent class. Just using `super()` will call the parent constructor

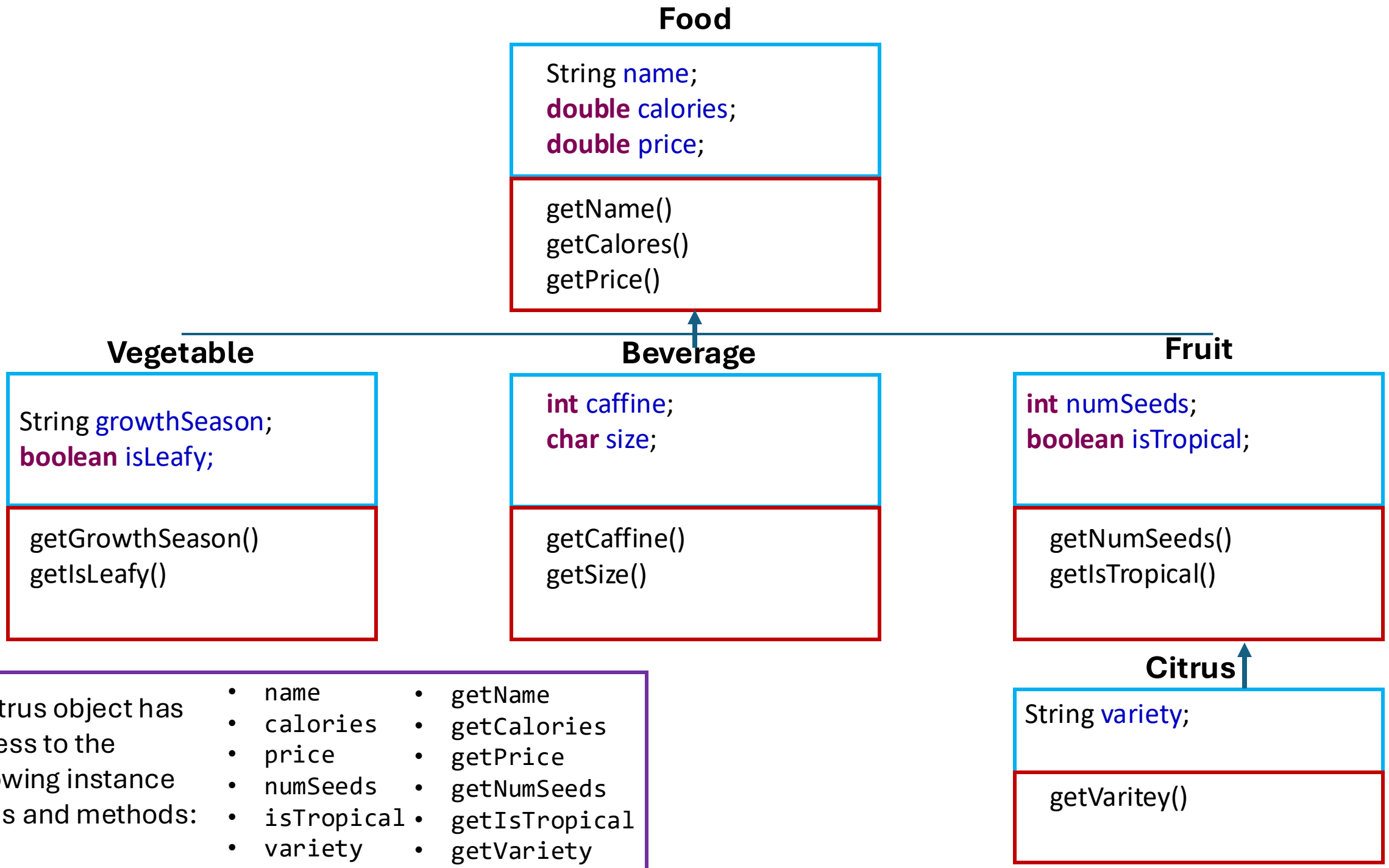
```
public abstract class Food {

}

public class Fruit extends Food {

}

public class Citrus extends Fruit {

}
```



In Java, we can only inherit from one class (but that one class we inherit from can also inherit from another class)

In this example, *`Citrus`* indirectly has access to the *`Food`* class instance fields/methods because the *`Fruit`* class inherits from *`Food`*


Food
String name;
double calories;
double price;
getName()
getCalores()
getPrice()
Vegetable
String growthSeason;
boolean isLeafy;
getGrowthSeason()
getIsLeafy()
Beverage
int caffine;
char size;
getCaffine()
getSize()
Fruit
int numSeeds;
boolean isTropical;
getNumSeeds()
getIsTropical()
Citrus
String variety;
getVaritey()


Food
String name;
double calories;
double price;
getName()
getCalores()
getPrice()
Vegetable
String growthSeason;
boolean isLeafy;
getGrowthSeason()
getIsLeafy()
Beverage
int caffine;
char size;
getCaffine()
getSize()
Fruit
int numSeeds;
boolean isTropical;
getNumSeeds()
getIsTropical()
Citrus
String variety;
getVaritey()
A Citrus object has access to the following instance fields and methods:
name
calories
price
numSeeds
isTropical
variety
getName
getCalories
getPrice
getNumSeeds
getIsTropical
getVariety

Method Precedence

Food.java

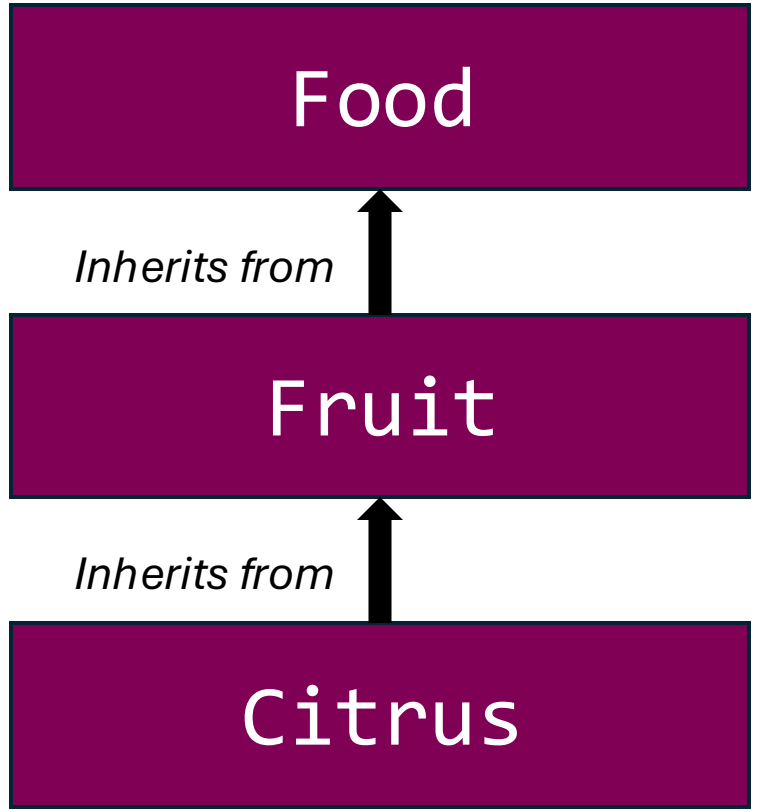
```
public String getName() {
        System.out.println(“Method #1 (Food)”);
}
```

Fruit.java

```
public String getName() {
        System.out.println(“Method #2 (Fruit)”);
}
```

Cirtus.java

```
public String getName() {
        System.out.println(“Method #3 (Citrus)”);
}
```



What if we define the exact same method in three different classes?

```
Citrus orange = new Citrus("Orange", 95, 1, 5, false, "Navel");
orange.getName()
```

What will get printed out?

Output:
`Method #3 (Citrus)`

Java will first look at the child class, and then move up the the parent classes

Method Precedence

Food.java

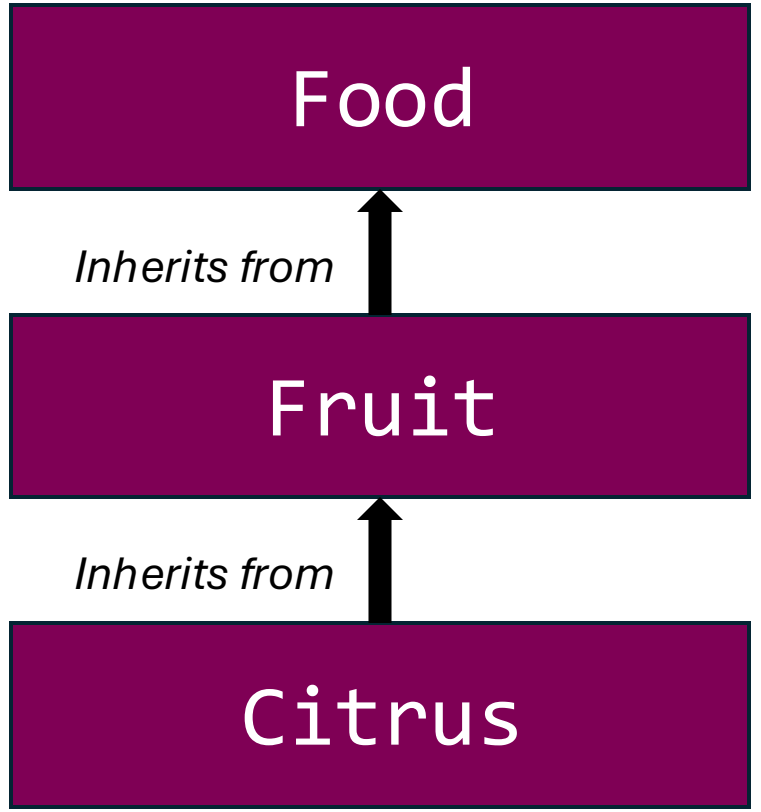
```
public String getName() {
        System.out.println("Method #1 (Food)");
}
```

Fruit.java

```
public String getName() {
        System.out.println("Method #2 (Fruit)");
}
```

Cirtus.java

(method deleted)



What if we define the exact same method in three different classes?

```
Citrus orange = new Citrus("Orange", 95, 1, 5, false, "Navel");
orange.getName()
```

What will get printed out?

Output:
`Method #2 (Fruit)`

Java will first look at the child class, and then move up the the parent classes