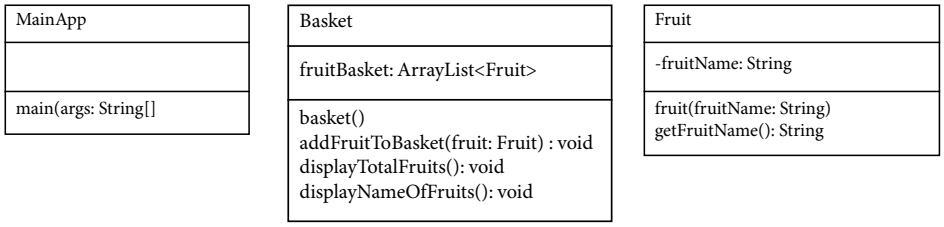
PART 5
JAVA: Visualize your OOP - by CodyScott
Para sa mga taga PHCORNER javadroids!

Ok, guys, let's go back to our 3 classes.
So far, we have:
1) MainAp - where we execute everything
2) Basket - where we can ADD fruits, COUNT fruits added, and also TELL the NAMES of the fruits.
3) Fruit - that we use to create or INSTANTIATE a Fruit OBJECT.

Cool! big deal... hehehehe



Let's add "FEATURES".

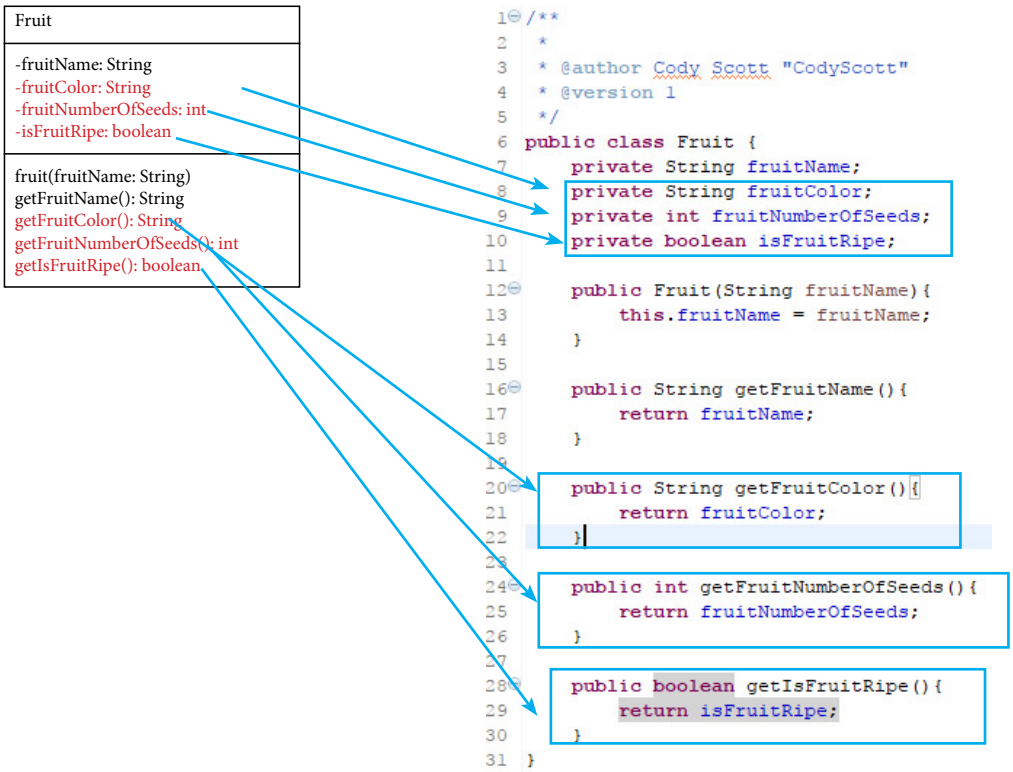
1) we want to know if the fruit is RIPE or NOT RIPE
2) we want to know the COLOR of the fruit
3) we want to know how many SEEDS the fruit has

....and again, kunyari lang na ang tatlong yan lang tayo interesado.

First question, RIPE, COLOR, SEEDS..... kaninong katangian ang mga ito? Basket or Fruits?
Obviously Fruits. So ida-dagdag natin ang mga katangian na ito sa Fruit Class

dinagdag natin yung mga katangian na gusto natin sa isang Fruit (attribute or variable)

dinagdag natin ang method or function kung paano natin hugutin ang mga katangian (getter methods)



NOTE: Bago nyo i-test, let's discuss JAVA Constructor. Very important.

JAVA Object Constructor

Keep in mind that a CLASS is just a blueprint. In order to use a CLASS, we need to CONSTRUCT an OBJECT from the CLASS
.....so CLASS---------is used to construct an--------OBJECT.
.....parang BLUEPRINT--------use to construct a---------HOUSE.

Sa code natin, we “constructed” Class ArrayList...and we also constructed Class Fruit and we used the syntax “new”.

```
import java.util.ArrayList;

/**
 *
 * @author Cody Scott "CodyScott"
 * @version 1
 */
public class Basket {
    ArrayList<Fruit> fruitBasket;

    public Basket(){
        fruitBasket = new ArrayList<Fruit>();
    }

    public void addFruitToBasket(Fruit fruit){
        fruitBasket.add(fruit);
    }

    public void displayTotalFruits(){
        System.out.println("Basket has " + fruitBasket.size() + " fruits");
    }

    public void displayNameOfFruits(){
        for(Fruit thisFruit: fruitBasket){
            System.out.println(thisFruit.getFruitName());
        }
    }
}
```

```
/**
 *
 * @author Cody Scott "CodyScott"
 * @version 1
 *
 */
public class MainApp {

    public static void main(String[] args) {
        //1. create a basket for the fruits
        Basket fruitBasket = new Basket();

        //2. create a fruit(s)
        Fruit apple = new Fruit("Apple");
        Fruit kaimito = new Fruit("Kaimito");
        Fruit rambutan = new Fruit("Rambutan");

        //3. let's add the fruits to the basket
        fruitBasket.addFruitToBasket(apple);
        fruitBasket.addFruitToBasket(rambutan);
        fruitBasket.addFruitToBasket(kaimito);

        //4. now, let's find out kung ilang fruits ang nasa basket
        fruitBasket.displayTotalFruits();

        //5. display names of fruits
        fruitBasket.displayNameOfFruits();
    }
}
```

So bale,

Fruit prutas1 = new Fruit(“Niyog”);

This is the name of the class

ang gusto nating vari-able name

and this is the CONSTRUCTOR na ginawa natin sa Fruit Class (meron parameter na String)

everytime na mag co-con-struct ng Fruit, gusto natin magbibigay sila ng pangalan ng agad agad.

```
1  /**
2   *
3   * @author Cody Scott "CodyScott"
4   * @version 1
5   */
6  public class Fruit {
7      private String fruitName;
8      private String fruitColor;
9      private int fruitNumberOfSeeds;
10     private boolean isFruitRipe;
11
12     public Fruit(String fruitName){
13         this.fruitName = fruitName;
14     }
15
16     public String getFruitName(){
17         return fruitName;
18     }
19
20     public String getFruitColor(){
21         return fruitColor;
22     }
23
24     public int getFruitNumberOfSeeds(){
25         return fruitNumberOfSeeds;
26     }
27
28     public boolean getIsFruitRipe(){
29         return isFruitRipe;
30     }
31 }
```

Kung gusto natin magbigay din sila ng fruitColor, fruitNumberOfSeeds....ang constructor code magiging;

```
public Fruit(String fruitName, String fruitColor, int fruitNumberOfSeeds){
        this.fruitName = fruitName;
        this.fruitColor = fruitColor;
        this.fruitNumberOfSeeds = fruitNumberOfSeeds;
}
```

refers to the Fruit variable

refers to the pa-rameter

So pag nag construct na ay ganito na:

Fruit prutas1 = new Fruit("Niyog", "Green", 1)

fruitName fruitColor fruitNumberOf-Seeds

Important: Kung paano ang sequence ng pag declare sa constructor, dapat ganon din ang sequence sa pag lagay ng parameter values. Hindi puwe-deng, (1, "Niyog", "Green")... mag e-error.

May buto ba ang niyo? hmmmm.... :)
You can put 0 kung wala.

IN GOOD JAVA PROGRAMMING PRACTICE

para mas flexible ang isang class, we DO NOT specify a constructor as much as possible.

so tanggalin natin yung construc-tor natin.

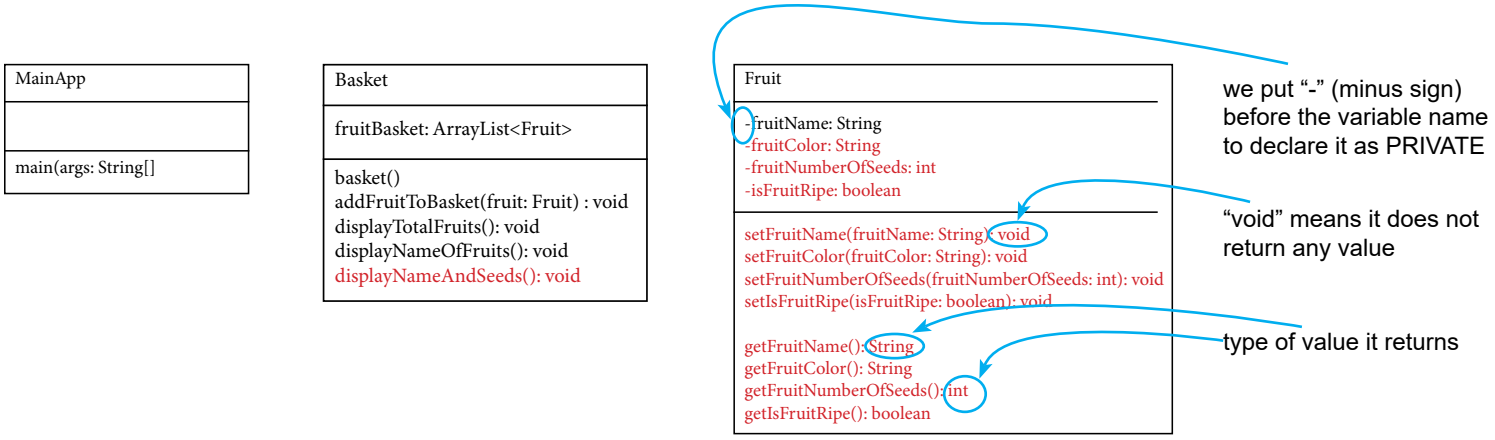
Instead, let's rely on the SETTERS and GETTERS

```
/**
 *
 * @author Cody Scott "CodyScott"
 * @version 1
 */
public class Fruit {
    private String fruitName;
    private String fruitColor;
    private int fruitNumberOfSeeds;
    private boolean isFruitRipe;

    //setters
    public void setFruitName(String fruitName){
        this.fruitName = fruitName;
    }

    public void setFruitColor(String fruitColor){
        this.fruitColor = fruitColor;
    }

    public void setFruitNumberOfSeeds(int fruitNumberOfSeeds){
        this.fruitNumberOfSeeds = fruitNumberOfSeeds;
    }

    public void setIsFruitRipe(boolean isFruitRipe){
        this.isFruitRipe = isFruitRipe;
    }

    //getters
    public String getFruitName(){
        return fruitName;
    }

    public String getFruitColor(){
        return fruitColor;
    }

    public int getFruitNumberOfSeeds(){
        return fruitNumberOfSeeds;
    }

    public boolean getIsFruitRipe(){
        return isFruitRipe;
    }
}
```

setter = sets/gives/assign a value to a variable when we construct
"A good point for Java. Dito lang puwede baguhin ang variable for security reason."

getter = gets/tells/provides the value of a variable when we ask it.
"A good point for Java. Dito lang puwede i-access ang variable for security reason."

So now, our UMLs look like these.



I hope you guys meron kayong napulot kahit konti.

Hanggang sa muli.