CMPSC 100

Computational Expression

Course announcements

- The quiz for today will be moved to next Monday (11 November)
- Group project proposal phase is now due Friday, November 8
  - Be sure to book time to meet with me regarding the feedback I gave to your group's idea list

Switch statements

“Switch” statements

```
boolean isCat = false;
String name = “Ulysses”;
switch (name) {
  case “Prof. Luman”:
    isCat = false;
    break;
  case “Ulysses”:
    isCat = true;
    break;
  default:
    isCat = false;
}
```

“If” statements

```
boolean isCat = false;
String name = “Ulysses”;
if (name.equals(“Prof. Luman”)) {
  isCat = false;
} else if (name.equals(“Ulysses”)) {
  isCat = true;
} else {
  isCat = false;
}
```

Switch statements

- Cannot resolve comparisons or boolean operations
  - Are tests against values *directly*, not their comparisons to other values
  - Switch statements are not necessarily “logical” tests
- Are expressed in a set of “cases” rather than “conditions”
- Requires definition of a “default” case for any values not specified

Do…while loops

```
String response = ‘n’;
do {
  System.out.println(“Do you want to end this loop [Y/N]? ”);
  response = input.nextLine();
} while (response.equalsIgnoreCase(‘N’));
```

```
String response = ‘n’;
while (response.equalsIgnoreCase(‘N’) {
  System.out.println(“Do you want to end this loop [Y/N]? “);
  response = input.nextLine();
}
```

Do…while loops

- Compares the “truth” value of the condition at the end of each iteration
  - Guarantees that the task in the loop runs *at least once*
- This is useful when testing user responses (e.g. if users have more input to enter)
- Beyond certain use cases, `do…while` is not as common as other forms of loops (`while`, `for`)

For loops

- Allows developer to initialize an identifier in the statement in order to track number of iterations, place in an ArrayList, et al.
- Test completely against the “truth” value of a single condition being met -- always an arithmetic test

For loops

“while” loop

```
int index = 0;
While (index < library.size()) {
  System.out.println(library.get(index);
  index++;
}
```

“for” loop

```
for (int i = 0; i < library.size() ; i++) {
    System.out.println(library.get(i));
}
```

Exercise

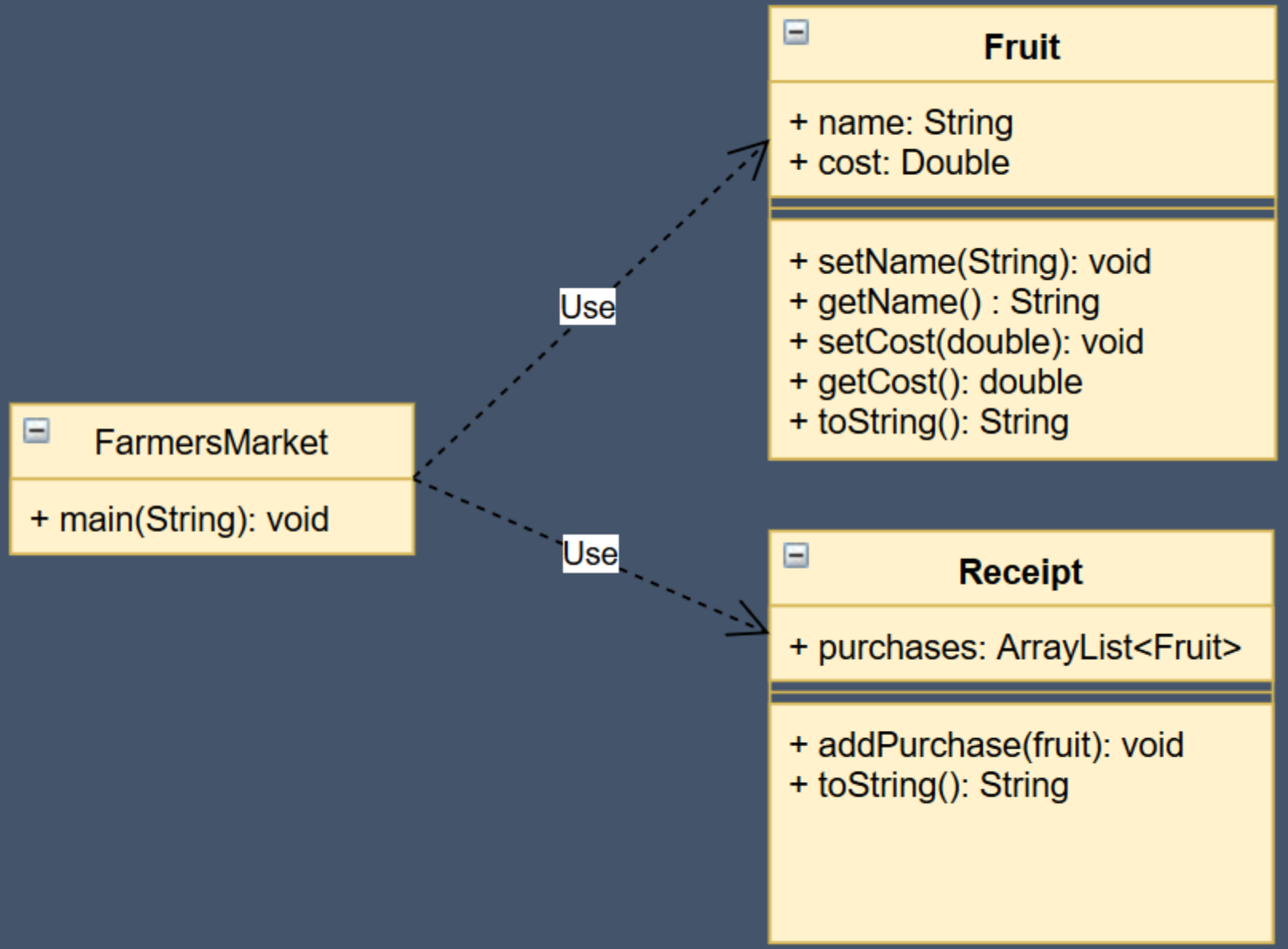
Navigate to the `class-activities/3-november` folder.

Exercise

We're going to the last farmer's market of the season, in what appears to be a rather temperate zone.

This market allows the user to:

- Start with a "fruit bankroll" from which to buy fruit
- Buy different varieties of fruit with a name and a cost
- Continue until either done buying or out of money
- Print the day's fruit bonanza as a kind of receipt listing the fruit purchased at the price at which it was purchased


Fruit
+ name: String
+ cost: Double
+ setName(String): void
+ getName() : String
+ setCost(double): void
+ getCost(): double
+ toString(): String
FarmersMarket
+ main(String): void
Use
Use
Receipt
+ purchases: ArrayList<Fruit>
+ addPurchase(fruit): void
+ toString(): String

```
public String toString() {
    String receipt = "Your purchases:\n";
    for(int i = 0; i < this.purchases.size(); i++){
      receipt += this.purchases.get(i) + "\n";
    }
    return receipt;
  }
```

RECEIPT.JAVA

```
Fruit fruit;
String choice;
String response;
double price = 0;
double bankroll = // Enter any number here
Receipt receipt = new Receipt();
Scanner input = new Scanner(System.in);
```

FARMERSMARKET.JAVA

```
do {
    System.out.println("You have $" + bankroll + " to purchase fruit.");
    System.out.print("What fruit would you like to purchase? ");
    choice = input.nextLine();
    fruit = new Fruit();
    /*
     * Space for switch statement
     */
    bankroll -= fruit.getCost();
    System.out.print("Would you like purchase more fruit? [Y/N]: ");
    response = input.nextLine();
  } while ((bankroll > price &&
          response.equalsIgnoreCase("Y"));
```

FARMERSMARKET.JAVA

```
switch(choice) {
  case "pear":
      fruit.setName(choice);
      fruit.setCost(1.00);
      receipt.addPurchase(fruit);
      break;
  case "apple":
      fruit.setName(choice);
      fruit.setCost(2.00);
      receipt.addPurchase(fruit);
      break;
  default:
      System.out.println(“Cannot process transaction”);
}
```

Insert two cases of your own!

FARMERSMARKET.JAVA

```
} while ((bankroll > price) &&
         response.equalsIgnoreCase(‘Y’));
if (bankroll < price) {
  System.out.println("You ran out of money for fruit! You bought:");
} else {
  System.out.println("Hope you're happy with your purchases:");
}
System.out.println(“------”);
System.out.println(receipt);
```

FARMERSMARKET.JAVA

Test using `gradle -q --console plain run`