CS 106A, Lecture 21
Classes

suggested reading:

Java Ch. 6
Plan for today

• Recap: HashMaps + What’s Trending
• Classes
• Recap
Recap: HashMaps

Tweets file: EllenTweets.txt
#tbt: 42
#findingdory: 20
#laughdancepartner: 55
#laughdancepartnerâ€™: 19
#edbypetsmart: 21
#littlebigshots: 18
#thebachelor: 16
#oscars: 19
#firstdates: 33
Plan for today

- Recap: HashMaps + What’s Trending
- Classes
- Recap
Large Java Programs

There are some *large* programs written in Java!
Defining New Variable Types

- Inbox Database
- Email Sender
- Login Manager
- Email
- User
- Inbox
A class defines a new variable type.
Why Is This Useful?

• A student registration system needs to store info about students, but Java has no **Student** variable type.

• A music synthesizer app might want to store information about different types of instruments, but Java has no **Instrument** variable type.

• An email program might have many emails that need to be stored, but Java has no **Email** variable type.

• **Classes** let you define new types of variables, which lets you decompose your program code across different files.
Classes Are Like Blueprints

iPod blueprint (class)

state:
current song
volume
battery life

behavior:
power on/off
change station/song
change volume
choose random song

constructs

iPod (variable) #1
state:
song = "1,000,000 Miles"
volume = 17
battery life = 2.5 hrs

behavior:
power on/off
change station/song
change volume
choose random song

iPod (variable) #2
state:
song = "Letting You"
volume = 9
battery life = 3.41 hrs

behavior:
power on/off
change station/song
change volume
choose random song

iPod (variable) #3
state:
song = "Discipline"
volume = 24
battery life = 1.8 hrs

behavior:
power on/off
change station/song
change volume
choose random song
A class defines a new variable type.
Creating A New Class

Let’s define a new variable type called **BankAccount** that represents information about a single person’s bank account.

A **BankAccount**:  
- contains the name of account holder  
- contains the balance  
- can deposit money  
- can withdraw money
What if if...

What if we could write a program like this:

```java
BankAccount nickAccount = new BankAccount();
nickAccount.setName("Nick");
nickAccount.deposit(50);

BankAccount rishiAccount = new BankAccount();
rishiAccount.setName("Rishi");
rishiAccount.deposit(50);
boolean success = rishiAccount.withdraw(10);
if (success) {
    println("Rishi withdrew $10.");
}
```
Creating A New Class

1. What information is inside this new variable type? These are its private instance variables.
Example: BankAccount

// In file BankAccount.java

public class BankAccount {
    // Step 1: the data inside a BankAccount
    private String name;
    private double balance;
}

Each BankAccount object has its own copy of all instance variables.
Creating A New Class

1. What information is inside this new variable type? These are its instance variables.

2. What can this new variable type do? These are its public methods.
What if we could write a program like this:

```java
BankAccount nickAccount = new BankAccount();
nickAccount.setName("Nick");
nickAccount.deposit(50);
println(nickAccount);

BankAccount rishiAccount = new BankAccount();
rishiAccount.setName("Rishi");
rishiAccount.deposit(50);
boolean success = rishiAccount.withdraw(10);
if (success) {
    println("Rishi withdrew $10.");
}
```
public class BankAccount {
    // Step 1: the data inside a BankAccount
    private String name;
    private double balance;

    // Step 2: the things a BankAccount can do
    public void deposit(double amount) {
        balance += amount;
    }

    public boolean withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            return true;
        }
        return false;
    }
}

Defining Methods In Classes

Methods defined in classes can be called on an instance of that class.

When one of these methods executes, it can reference that object’s copy of instance variables.

```java
ba1.deposit(0.20);
ba2.deposit(1000.00);
```

This means calling one of these methods on different objects has different effects.
Getters and Setters

Instance variables in a class should *always be private*. This is so only the object itself can modify them, and no-one else.

To allow the client to reference them, we define public methods in the class that *set* an instance variable’s value and *get* (return) an instance variable’s value. These are commonly known as *getters* and *setters*.

```java
account.setName("Nick");
String accountName = account.getName();
```

Getters and setters prevent instance variables from being tampered with.
public class BankAccount {
    private String name;
    private double balance;

    ...

    public void setName(String newName) {
        if (newName.length() > 0) {
            name = newName;
        }
    }

    public String getName() {
        return name;
    }
}
Creating A New Class

1. What information is inside this new variable type? These are its instance variables.

2. What can this new variable type do? These are its public methods.

3. How do you create a variable of this type? This is the constructor.
Constructors

GRect rect = new GRect();

GRect rect2 = new GRect(50, 50);

This is calling a special method! The GRect constructor.
Constructors

BankAccount bal = new BankAccount();

BankAccount ba2 = new BankAccount("Nick", 50);

The constructor is executed when a new object is created.
public class BankAccount {
    // Step 1: the data inside a BankAccount
    private String name;
    private double balance;

    // Step 2: the things a BankAccount can do (omitted)
    // Step 3: how to create a BankAccount
    public BankAccount(String accountName, double startBalance) {
        name = accountName;
        balance = startBalance;
    }

    public BankAccount(String accountName) {
        name = accountName;
        balance = 0;
    }
}
Using Constructors

BankAccount ba1 =
    new BankAccount("Marty", 1.25);

BankAccount ba2 =
    new BankAccount("Mehran", 900000.00);

- When you call a constructor (with **new**):
  - Java creates a new object of that class.
  - The constructor runs, on that new object.
  - The newly created object is returned to your program.
Constructors

• **constructor**: Initializes the state of new objects as they are created.

```java
public ClassName(parameters) {
    statements;
}
```

– The constructor runs when the client says `new ClassName(...)``;

– no return type is specified; it "returns" the new object being created

– If a class has no constructor, Java gives it a *default constructor* with no parameters that sets all fields to default values like 0 or null.
Plan for today

• Recap: HashMaps + What’s Trending
• Classes
• Recap
What Is A Class?

A class defines a new variable type.
Creating A New Class

1. What information is inside this new **variable type**? These are its instance variables.

2. What can this new variable type do? These are its public methods.

3. How do you create a variable of this type? This is the constructor.
Recap

• Recap: HashMaps + What’s Trending
• Classes
• Recap

Next time: more classes