Object-Oriented Programming in Java
IFT 194: Lab 3

Brandon Doyle
bdoyle5@asu.edu
1215232174

Dr. Usha Jagannathan
Usha.Jagannathan@asu.edu

July 15, 2018

Summary

Prelab Exercises

1. Class constructors are special methods that are called when the `new` operator is used to create a new instance of a class. These methods typically set up attributes of the class.

   There are a few differences between regular methods and class constructors (special methods). First and foremost, constructors cannot have a return value, and neither is a return type specified in the method header. Secondly, constructors have the same name as the containing class. Thus, if I were to write a class `Animal`, this class' constructors would also have the name `Animal`.

2. Access or visibility modifiers, which include `public` and `private`, among others, are used to determine where a variable or method of a class may be accessed from. For example, a public method or field variable may be accessed from outside an instance, whereas in the latter case they may only be used from inside the instance.

   A programmer can decide if a variable should be public based on what that variable may store. Likewise, a programmer may make a method public if it plays a vital role in the class' interface to other classes. Private methods are typically used to provide support to public methods. We should also keep *encapsulation* in mind, which suggests that a class should not allow other classes to modify its state by "reaching in" and modifying a value. Rather, we should provide an interface, perhaps through a public method, that has the ability to modify its state. This being said, it is perfectly fine to define publicly-accessible constants as long as they are preceeded by `final`, declaring the value immutable.

   There are a lot of best-practices suggested throughout the text for this course. I completely support them, knowing that things can easily get out of hand as a code base becomes large enough that a single person can no longer maintain or extend it.

3. In this problem we consider a class that may represent a bank account.

   a. To hold information about an account balance, the name of the account holder, and an account number, I might define the following field variables.

   ```
   private double balance = 0.0;
   final private String accountHolder;
   final private int[] accountNumber;
   ```

   The `accountHolder` and `accountNumber` fields are declared `final` because I don't want them to be modifiable once the class is instantiated. Moreover, all variables are private, because I wouldn't want this information to be accessible by other classes or objects unless the proper indentification is provided or process completed.

   b. In this sub-problem, we're asked to write method headers for each of the examples provided.

      i. To withdraw a certain amount of money from the account – change the account balance and do not return a value.

      ```
      public void withdraw(double amount) { ... }
      ```

View the source of this document on GitHub.

ii. Deposit a certain amount into the account and do not return a value.

```
public void deposit(double amount) { ... }
```

iii. Get the current balance of the account.

```
public double getBalance(/* something to check credentials? */) { ... }
```

iv. Return a string with account information, including name, account number, and balance.

```
public String getAccountInfo(/* Again, check credentials? */) { ... }
```

v. Charge a $10 fee.

```
public void chargeFee(double amount) { ... }
```

vi. A constructor that requires the initial balance, name of the owner, and account number.

```
public BankAccount(double initialBalance, String owner, int[] acctNumber) { ... }
```

Bank Account Class

1. For this question, I've reproduced the code provided in the lab in **??**.

Code

Tracking Grades

Band Booster Class

Representing Names