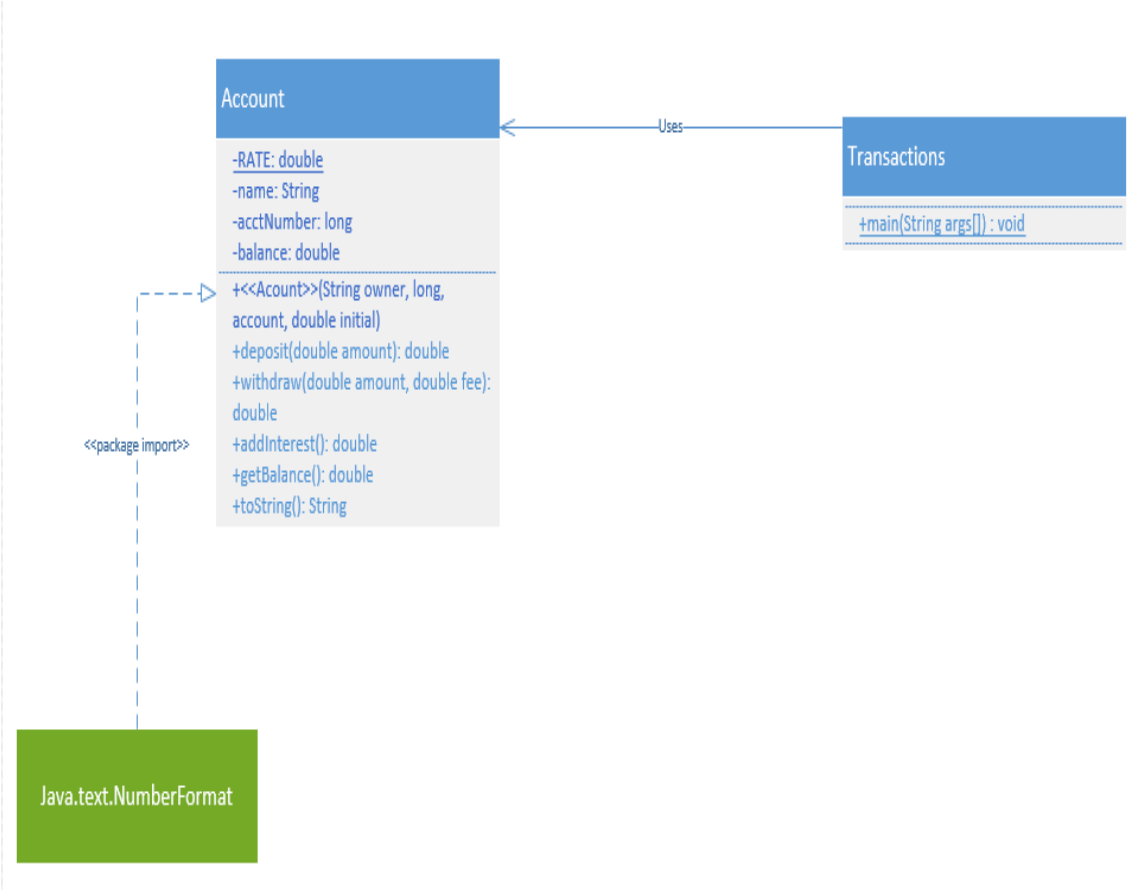
Matthew Vollkommer
8102494122
HW 2

1) It does the code written in the try block. If an exception is thrown, it catches the exception. It is good because the program does not end when an exception is caught. It is good because Java does not end the program based on an error, Marty has taken control of the situation. It is good because all exceptions would be caught. It is not good because it does nothing to try to rectify the thrown exception. Instead it continues running the same way it would run if no exception were thrown. This is bad because it may lead to bugs later in the program that would be difficult to track back to this specific block of code. It is bad because no exception is thrown. The person running the program will not know an error has occurred. It also does not differentiate between different types of exceptions.

2) The code is well indented and easy to read. He used dynamic variables in his methods instead of completely static code. He also makes a decent attempt to catch errors. However, Biff failed to comment his code, making interpretation more difficult than it needs to be. The String outputFileName is passed into the method parameters. He then creates a new Formatter object called outputFile. The ouputFileName is passed to the Formatter object. The Formatter object, outputFile is dependent on finding a file of the same name as the outputFileName string. If the file is not found, the exception is caught. If no file is found. It prints out an appropriate error message. If a security exception is thrown, it catches the SecurityException and prints out an appropriate error message. Everytime the code is ran, the finally block runs. No matter what happens, the he saves the buffered data files through the use of the infile class's close() method. The system then exits. This is bad because if one of the previous errors occurs, a new error may occur at the infile.close() statement. It is also bad because the program will always end when this method runs at the System.exit(1) statement. Marty should not have this code in the finally block. He should remove the finally block altogether. He should place the system.exit(1) statement after he outputs the information in the caught exception blocks. He should only run the close method of the infile class if he has successfully imported a file.

3) A static method cannot refer to an instance variable. For example, the main method is a static method. The main method can be called without instantiating the class, so there was no object of the public class created; there is no instance variable to refer to.

2


Account
-RATE: double
-name: String
-acctNumber: long
-balance: double
+<<Acount>>(String owner, long, account, double initial)
+deposit(double amount): double
+withdraw(double amount, double fee): double
+addInterest(): double
+getBalance(): double
+toString(): String
Uses
Transactions
+main(String args[]) : void
<<package import>>
Java.text.NumberFormat

4)

3

5)


Transaction
{Abstract}
-cost: douhle
-transNumber: long
+<<Transaction>>(double cost, long transNum)
+purchase(double amount): double
+return(double amount): double
Is a type of transaction
IS A
Is a type of Transaction
Cash
-amount: double
+setAmount(double amount): void
+getAmount(): double
Card
{Abstract}
-cardName: String
-Name: String
-cardCompany: String
-cardNum: long
+<<Constructor>>(String cardName, String ownerName, String cardCompany, String cardNum)
+ setCardName(String name): void
+setCardCompany(String company): void
+setCardNum(long cardNum): void
+getCardName(): String
+getCardNum(): long
GiftCard
-storeName: String
-Amount: double
+setStoreName(String storeName): void
+getStoreName(): string
+getAmount(): double
+setAmount(double amount): void
Is a Type of Card
Is A
Is a Type of Card
BankCard
{Abstract}
-type: String
+setType(String cardType): void
+getType() String
Uses
Is A bankCard
Is a BankCard
Is A
<<package import>>
Account
-RATE: double
-name: String
-acctNumber: long
-balance: double
+<<Acount>>(String owner, long, account, double initial)
+deposit(double amount): double
+withdraw(double amount, double fee): double
+addInterest(): double
+getBalance(): double
+toString(): String
Java.text.NumberFormat
DebitCard
-debitLimit: double
+setDebitLimit(double debitLimit): void
+getDebitLimit(): double
CreditCard
-creditLimit: double
+setCreditLimit(double creditLimit): void
+getCreditLimit(): double

Polymorphism is the ability to present the same interface for differing underlying objects.

In the hierarchy shown above, there are many different types of transaction objects from cash to Credit card. The specific object used for the transaction is irrelevant as long as the transaction works. At the POS, the two parties only care about the immediate affects that take place ie(the methods and instance variables in the transaction class). With polymorphism, all the children of the Transaction parent class maintain the methods and instance variables of the Transaction class, regardless of how far down the hierarchy the child is. Debit cards, credit cards, cash, and gift cards are able to maintain methods and instance variables, without the need to repeat the code of their parents.

6)

A)

Beginning HW2 - Part I
Two's Constructor
Get it?
One World
Get it?

Beginning HW2 - Part II
Two's Constructor
One World

Beginning HW2 – Part III
Two's Constructor
One World

Beginning HW2 – Part IV

B)

```
Beginning HW2 – Part I
One's Constructor
Two's Constructor
Get it?
One World
Get it?

Beginning HW2 – Part II
One's Constructor
Two's Constructor
Get it?

Beginning HW2 – Part III
One's Constructor
Two's Constructor
```

```
Get it?

Beginning HW2 – Part IV
```

I was not expecting the line labeled QC to ouput what it did. The line at QC typecasted anotherObject to an object of the parent class One. I therefore expected to see the String “One World” instead, it output the String “Get it?” The String from the initial child class overrode the String from the Parent class.

C) Throughout the entire assignment, I did not print out “One’s Constructor” before the child class was constructed, the parent class was constructed. I thought the parent constructor would have been ignored altogether. The parent constructor overloads the child constructor. A parent must be constructed before its child.

On part II I printed “One World” instead of “Get It?”. I thought it would print the string from the One class. However, because the object was set equal to an object of the Two class, the String from the Two class, the child class overrode the String from the parent class, the One Class.

D) I get the following compile time error:

```
-bash-4.1$ make
javac *.java
HW2.java:15: error: cannot find symbol
        anObject.print2();
                 ^
  symbol:   method print2()
  location: variable anObject of type One
1 error
make: *** [compile] Error 1
```

I would typecast anObject from an Object of the One class to an Object of the Two class

```
      ((Two)anObject).print2();
```