Lab14: BankAccount class

Day 1

Part 1: Create a BankAccount class along with private instance variables:
double balance, String name

```
public class BankAccount() {
  // instance variables
  private String nane;
  private double balance;
}
```

Question 1: When can you access private instance variables?

Only code written in the side of the BankAccount class can see these private variables/methods

Part 2: Create getters and setters for name and balance.

```
// gets the name/balance
public String getName () {
  return name;
}

public double getBalance () {
  return balance;
}

// sets the name/balance to the variable
public void setName (String newName) {
  name = newName;
}

public void setBalance (double newBalance) {
  balance = newBalance
}
```

Question 2: A programmer decides that the account's name should not be changeable. How would you accomplish this?

It removes access by setting the setter as a private method. IT could also be deleted but using private would not suffice.

Part 3: In Main.main(), create BankAccounts for the following:
a. Al has $50.
b. Bob has $25.
c. Carl has $44.

```
// declaration of BankAccount object
// modifying of the BankAccount object with a String and double value

// a) Al's Account
BankAccount alAccount = new BankAccount();
alAccount .setName("Al");
alAccount .setBalance(50.00);

// b) Bob's Account
BankAccount bobAccount = new BankAccount();
bobAccount .setName("Bob");
bobAccount .setBalance(25.00);

// c) Carl's Account
BankAccount carlAccount = new BankAccount();
carlAccount .setName("Carl");
carlAccount .setBalance(44.00);
```

Lab14: BankAccount class

Day 2

Part 4: Create the following instance and class variables:
public int accountNumber; // Do not initialize instance variables.
public static int numberOfAccountsCreated = 0; // Do initialize class/static variables.

```
// creation of instance variables for accountNumber & numberOfAccountsCreated
public int accountNumber;
public static int numberOfAccountsCreated = 0;
```

Question 3: What would be the consequence of making the account's name a static variable?

It would not allow modification in the main class.

Part 5: Now that we have shown how instance/class variables work, go back and make them both private. Create getters for both, but not setters. Paste your getters below.

```
private static int getAccountNumber () {
  // a getter for the accountNumber
  return accountNumber;
}

private static int getnumberOfAccountsCreated() {
  // getter for the numberOfAccountsCreated
  return numberOfAccountsCreated;
}
```

Part 6: Create an empty constructor. The accountNumber should be 10000000 + the number of accounts created. So the first BankAccount should be 10000001, the second BankAccount should be 10000002. The header for the empty constructor should be:

```
public BankAccount();
```

You should handle the accountNumber and accountsCreated yourself.

```
public BankAccount () {
  // begin constructor
  setName("");
  name = "";

  // increment the numberOfAccountsCreated by one
  numberOfAccountsCreated++;

  // add to dind the total accountNumber
  accountNumber = 10000000 + numberOfAccountsCreated;
}
```

Create a full constructor. The accountNumber should be 10000000 + the number of accounts created. So the first BankAccount should be 10000001, the second BankAccount should be 10000002. The header for the full constructor should be:

```
public BankAccount(String n, double b);
```

Lab14: BankAccount class

You should handle the accountNumber and accountsCreated yourself.

```
public BankAccount (String name, double balance) {
  // sets the name and balance
  setName(name);
  setBalance(balance);
  numberOfAccountsCreated++;

  // increment number of accounts created
  accountNumber = 10000000 + numberOfAccountsCreated
}
```

Part 7: Create a toString() method in the BankAccount class that returns a String such as "Al's account 10000001 has a balance of $50". This method uses data specific to each account. Should the account be static or non-static?

```
public String toString() {
  return name + "'s bank account " + getAccountNumber();
}
```

Day 3

Part 8: Modify the constructors so that every 100th account gets a $100 bonus. So if you open the 100th, 200th, 300th, etc BankAccount with $12, your balance should start as $112.
Create an array of 500 BankAccounts to test, with account names Acct1, Acct2, Acct3, etc.
Each BankAccount should have a balance of $1, $2, ... $500.
Print out all 500 BankAccounts.

```
public BankAccount () {
  setName("");
  name = "";
  numberOfAccountsCreated++;
  accountNumber = 10000000 + numberOfAccountsCreated;

  if (accountNumber % 100 == 0) { // if at the 100th
    bankaccount balance += 100.0;
  }
}

public BankAccount (String n, double b) {
  setName(n);
  numberOfAccountsCreated++;
  accountNumber = 10000000 + numberOfAccountsCreated;

  // if it's at the 100th bankAccount
  if (numberOfAccountsCreated % 100 == 0) {
    setBalance(100 + b); // add 100 to balance
  }

  // if not, it won't change the value
  else {
    setBalance(b);
  }

}

Main:

BankAccount[] a = new BankAccount[500];
for (int i = 0; i < 500; i++) {
  a[i] = new BankAccount("Account" + (i + 1), i + 1);
  System.out.println(a[i].toString());
```

```
}
```

Part 9: In the BankAccount class, create a deposit method which adds money to the account balance. Your deposit method should only allow positive values to be deposited. Print a message, "Invalid deposit amount." otherwise. Create a withdraw method that returns a double representing the amount of money withdrawn and subtracts the value from the balance. If the withdrawl amount is bigger than the balance, do not withdraw any money, return 0, and subtract $10 from the balance representing a bank fee.

```
public void deposit (double money) {
  // checks if the inputted amount is invalid
  if (money <= 0) {
    System.out.println("Invalid deposit amount.");
    return;
  }

  else {
    balance += money;
    System.out.println(money + "successfully deposited. Balance is now " +
    balance);
  }
}

public double withdraw (double withdrawlAmount) {
  if (withdrawlAmount < balance) {
    balance -= 10; // removes 10 from the user's balance return 0;
  }

  else {
    balance -= withdrawlAmount;
  }
}
```

Part 10: In Main.main(), create a bank app that has the menu:

```
1. Create new account
2. View all accounts
3. Quit
Enter your selection:
```

Lab14: BankAccount class

Your bank app should repeat until the user enters 3 to quit. IE, this bank app should be able to create multiple accounts in a row, print them all, then enter more accounts. You may assume that the user will create at most 100 accounts.

```
import java.util.Scanner;

public class Main {
  // code written in the side of the BankAccount class can access the private variables/methods

  public static void main(String[] args) {
    Scanner scannerObject = new Scanner(System.in);
    BankAccount [] account = new BankAccount[100];
    int userin = 0;

    while (userin < 3) {System.out.println("\nMenu");
      System.out.println("1) Create a new bank account");
      System.out.println("2) View all bank accounts");
      System.out.println("3) Quit");
      System.out.println("Selection: ");
      int re = input.nextInt();

      if (res == 1) {
        System.out.println("You chose option 1");
        input.nextLine();
        System.out.println("Name?");
        String name = input.nextLine();
        System.out.println("Initial desposit?");
        double d = input.nextDouble();

        if (d <=0 ) {
          continue;
        }

        account[userin] = new BankAccount(name, d);
        userin++;
        System.out.println("Account(s) successfully created");
      }

      else if {
        for (int i = 0; i < BankAccount.getNumberOfAccountsCreated(); i++) {
          System.out.println(account[i].getName() + ": " + account[i].getBalance());
        }
      }

      else {
        System.out.println("Thank you for visiting, goodbye!");
        break;
      }
    } // end of while loop
  } // end of main(Strin[] args)
} // end of Main
```