```
/*  Remember that you use the "this" keyword to show that you are referring to an
    instance variable.  You'll notice that throughout this class I switch on and off
    from using the "this" keyword.  I do this to show that you don't always have to
    use the "this" keyword to identify instance variables.  If you decide to use them
    or not use it is up to you, just be consistent.
*/

public class BankAccount {

    private String myPassword;
    private double myBalance;
    public static final double OVERDRAWN_PENALTY = 20.00;


    //Constructors
    /* Default constructor constructs bank account with default values. */
    public BankAccount() {
        myPassword = "";
        myBalance = 0.0;
    }


    /* Constructs bank account with specified password and balance. */
    public BankAccount(String myPassword, double balance) {
        this.myPassword = myPassword;
        this.myBalance = balance;
    }

    //Accessors
    /* Returns balance of this account.  */
    public double getBalance() {
        return this.myBalance;
    }

    /* Returns password of this account.  Notice I do not use the "this" keyword.
       Check with your company's policy */
    public String getPassword() {
        return myPassword;
    }


    //Mutator Methods
    /* Deposits amount in bank account with given password */
    public void deposit(String password, double amount) {
        if(!password.equals(myPassword)) {
             System.out.println("Incorrect Password.  Deposit Declined!");
        }else {
            myBalance += amount;
        }
    }

     /* Withdraws amount from bank account with given password
        Assesses penalty if myBalance is less than amount  */
     public void withdraw(String password, double amount) {
        if(!password.equals(myPassword)) {
             System.out.println("Incorrect Password.  Deposit Declined!");
        }else {
             myBalance -= amount;     //allows negatice balance
        }
        if (myBalance < 0) {
             myBalance -= BankAccount.OVERDRAWN_PENALTY;
        }
    }
}
```

```
ublic class BankAccountTest {

   public static void main (String[] args) {
       BankAccount a = new BankAccount();
       BankAccount b = new BankAccount("billS", 500.00);
       b.deposit("billS", 200);
       System.out.println("Bill has $" + b.getBalance() + " in his account");

   }
```

Handout B #1

This chooseBestAccount method attempts - erroneously - to set its betterFund parameter to the BankAccount with the higher balance:

```
    public static void chooseBestAccount(BankAccount better, BankAccount b1, BankAccount b2)
    {
        if(b1.getBalance() > b2.getBalance()) {
            better = b1;
        } else {
            better = b2;
        }
    }

    public static void main(String[] args) {
        BankAccount briansFund = new BankAccount("brian", 10000);
        BankAccount suesFund = new BankAccount("sue", 90000);
        BankAccount betterFund = null;

        BankAccount.chooseBestAccount(betterFund, briansFund, suesfund);
            ...
    }
```

Handout B #2

The way to fix the problem is to modify the method so that it returns the better account. Returning an object from a method means that you are returning the address of the object.

```
public static void chooseBestAccount(BankAccount b1, BankAccount b2) {
        BankAccount better;
        if(b1.getBalance() > b2.getBalance()) {
            better = b1;
        } else {
            better = b2;
        }
        return better;
}

public static void main(String[] args) {
        BankAccount briansFund = new BankAccount("brian", 10000);
        BankAccount suesFund = new BankAccount("sue", 90000);

        BankAccount betterFund = BankAccount.chooseBestAccount(briansFund, suesFund);
            ...
}
```

Handout B #3

What the previous method does not due is create a new object to which betterFund refers. To do
that would require the keyword new and use of a BankAccount constructor. Assuming that a getPassword() accessor has been added to the BankAccount class, the code would like this.

```
public static BamkAccount chooseBestAccount(BankAccount b1, BankAccount b2) {
        BankAccount better;
        if(b1.getBalance() > b2.getBalance()) {
            better = new BankAccount(b1.getPassword(), b1.getBalance());
        } else {
            better = new BankAccount(b2.getPassword(), b2.getBalance());
        }
        return better;
}

public static void main(String[] args) {
        BankAccount briansFund = new BankAccount("brian", 10000);
        BankAccount suesFund = new BankAccount("sue", 90000);


        BankAccount betterFund = BankAccount.chooseBestAccount(briansFund, suesFund);
            ...

}
```

Handout C #1

This method changes the state of the object to which the parameter refers through methods that act on the object.

```
/* Subtracts fee from balance in b if current balance is too low */
public static void chargeFee(BankAccount b, String password, double fee) {

    final double MIN_BALANCE = 10.00;

    if (b.getBalance() < MIN_BALANCE) {
        b.withdraw(password, fee);
    }
}

public static void main(String[] args) {

    final double FEE = 5.00;
    BankAccount andysAccount = new BankAccount("andy", 7.00);
    chargeFee(andysAccount, "andy", FEE);
}
```