## BJP5 Exercise 9.2: Janitor

**Language/Type:** Java classes implementing inheritance instance methods
**Related Links:** Employee.java
**Author:** Marty Stepp (on 2019/09/19)

Write a class `Janitor` to accompany the other law firm classes described in this chapter. Janitors work twice as many week as other employees (80 hours/week), they make $30,000 ($10,000 less than general employees), they get half as as other employees (only 5 days), and they have an additional method clean that prints `"Workin' for the man."` N interact with the superclass as appropriate.

**Type your solution here:**

```java
class Janitor extends Employee{
    public int getHours(){
        return (super.getHours() * 2);
    }

    public double getSalary(){
        return (super.getSalary() - 10000);
    }

    public int getVacationDays(){
        return (super.getVacationDays()/2);
    }

    public void clean(){
        System.out.println("Workin' for the man.");
    }
}
```

This is an **inheritance problem.** Write a Java class using inheritance. (You do not need to write any `import` statements.)

Submit

**You passed 7 of 7 tests.**

Go to the next problem: HarvardLawyer

**test #1:** `getSalary 1`

## BJP5 Exercise 9.3: HarvardLawyer

**Language/Type:** Java classes implementing inheritance instance methods
**Related Links:** Lawyer.java
**Author:** Marty Stepp (on 2019/09/19)

Write a class `HarvardLawyer` to accompany the other law firm classes described in this chapter. Harvard lawyer lawyers, but they make 20% more money than a normal lawyer, they get 3 days more vacation, and they have to fi lawyer's forms to go on vacation. That is, the `getVacationForm` method should return `"pinkpinkpinkpink"`. interact with the superclass as appropriate.

**Type your solution here:**

```java
class HarvardLawyer extends Lawyer{
    public double getSalary(){
        return (super.getSalary() * 1.2);
    }

    public int getVacationDays(){
        return (super.getVacationDays() + 3);
    }

    public String getVacationForm(){
        String form = "";
        for (int i=0; i<4; i++)
            form += (super.getVacationForm());
        return form;
    }

}
```

This is an **inheritance problem.** Write a Java class using inheritance. (You do not need to write any `import` statements.)

Submit

**You passed 7 of 7 tests.**

Go to the next problem: MonsterTruck

**test #1:** `getSalary 1`

👁 Show He

## ◯ BJP5 Exercise 9.9: MinMaxAccount

| | |
|---|---|
| **Language/Type:** | Java classes implementing inheritance instance methods |
| **Related Links:** | BankingAccount.java |
| **Author:** | Robert Baxter (on 2019/09/19) |

A company has written a large class `BankingAccount` with many methods including:

| Method/Constructor | Description |
|---|---|
| `public BankingAccount(Startup s)` | constructs a `BankingAccount` object using information in the `Startup` object *s* |
| `public void debit(Debit d)` | records the given debit |
| `public void credit(Credit c)` | records the given credit |
| `public int getBalance()` | returns current balance in pennies |

Design a new class `MinMaxAccount` whose instances can be used in place of a `BankingAccount` object but include new beh of remembering the minimum and maximum balances ever recorded for the account. You should provide the same methods as superclass, as well as the following new behavior:

| Method/Constructor | Description |
|---|---|
| `public MinMaxAccount(Startup s)` | constructs a `MinMaxAccount` object using information in the `Startup` object *s* |
| `public int getMin()` | returns minimum balance in pennies |
| `public int getMax()` | returns maximum balance in pennies |

The account's constructor sets the initial balance based on the `Startup` information. Assume that only the `debit` and `credit` methods change an account's balance.

**Type your solution here:**

```java
class MinMaxAccount extends BankingAccount{
    private int min;
    private int max;

    public MinMaxAccount(Startup s){
        super(s);
        min = getBalance();
        max = getBalance();
    }

    public void debit(Debit d){
        super.debit(d);
        newExtremes();
    }

    public void credit(Credit c){
        super.credit(c);
        newExtremes();
    }

    public void newExtremes() {
        int balance = getBalance();
        if (balance < min) {
            min = balance;
        } else if (balance > max) {
            max = balance;
        }
    }

    public int getMin(){
        return min;
    }

    public int getMax(){
        return max;
    }

}
```

This is an **inheritance problem.** Write a Java class using inheritance. (You do not need to write any `import` statements.)

4

**Submit**

☑ Sounc
☑ Highli

**✓ You passed 4 of 4 tests.**

Go to the next problem: DiscountBill

test #1: MinMaxAccount(s)

👁 Show Header

## ○ BJP5 Exercise 9.10: DiscountBill

| | |
|---|---|
| **Language/Type:** | Java classes implementing inheritance instance methods |
| **Related Links:** | GroceryBill.java |
| **Author:** | Robert Baxter (on 2019/09/19) |

Suppose a class `GroceryBill` keeps track of a list of items being purchased at a market:

| Method/Constructor | Description |
|---|---|
| `public GroceryBill(Employee clerk)` | constructs a `GroceryBill` object for the given `clerk` |
| `public void add(Item i)` | adds `i` to this bill's total |
| `public double getTotal()` | returns the cost of these items |
| `public void printReceipt()` | prints a list of items |

`GroceryBill` objects interact with `Item` objects. An `Item` has the following public methods:

| Method/Constructor | Description |
|---|---|
| `public double getPrice()` | returns the price for this item |
| `public double getDiscount()` | returns the discount for this item |

For example, a candy bar item might cost `1.35` with a discount of `0.25` for preferred customers, meaning that preferred customers get it for `1.10`. (Some items will have no discount, `0.0`.) Currently the above classes do not consider discounts. Every item in a bill is charged full price, and item discounts are ignored.

Define a class `DiscountBill` that extends `GroceryBill` to compute discounts for preferred customers. The constructor for `DiscountBill` accepts a parameter for whether the customer should get the discount.

Your class should adjust the amount reported by `getTotal` for preferred customers. For example, if the total would have been `$80` but a preferred customer is getting `$20` in discounts, then `getTotal` should report the total as `$60` for that customer. You should also keep track of how many items a customer is getting a non-zero discount for and the overall discount, both as a total amount and as a percentage of the original bill. Include the extra methods below that allow a client to ask about the discount:

| Method/Constructor | Description |
|---|---|
| `public DiscountBill(Employee clerk, boolean preferred)` | constructs discount bill for given `clerk` |
| `public int getDiscountCount()` | returns the number of items that were discounted, if any |
| `public double getDiscountAmount()` | returns the total discount for this list of items, if any |
| `public double getDiscountPercent()` | returns the percent of the total discount as a percent of what the total would have been otherwise |

If the customer is not a preferred customer the `DiscountBill` behaves at all times as if there is a total discount of 0.0 and no items have been discounted.

**Type your solution here:**

```java
class DiscountBill extends GroceryBill{
    private boolean preferred;
    private int discountItems;
    private double discountAmount;

//Constructor
    public DiscountBill(Employee clerk, boolean preferred){
        super(clerk);
        this.preferred = preferred;
        discountItems = 0;
        discountAmount = 0.0;
    }

//Methods
    public void add(Item item) {
        super.add(item);
        if (preferred && item.getDiscount() > 0) {
            discountItems++;
            discountAmount += item.getDiscount();
        }
    }

//Getter methods
    public double getTotal() {
        return (super.getTotal() - discountAmount);
    }


    public int getDiscountCount(){
        return discountItems;
    }

    public double getDiscountAmount(){
        return discountAmount;
    }

    public double getDiscountPercent(){
        return (discountAmount * 100 / super.getTotal());
    }






}
```

This is an **inheritance problem.** Write a Java class using inheritance. (You do not need to write any `import` statements.)

4 Indent

☑ Sound F/X
☑ Highlighting

Submit

You passed 5 of 5 tests.

Go to the next problem: FilteredAccount

Show Hea

## ○ BJP5 Exercise 9.11: FilteredAccount

| | |
|---|---|
| **Language/Type:** | Java classes implementing inheritance instance methods |
| **Related Links:** | Account.java |
| **Author:** | Eric Spishak (on 2019/09/19) |

A cash processing company has a class called `Account` used to process transactions:

| Method/Constructor | Description |
|---|---|
| `public Account(Client c)` | constructs an account using client information |
| `public boolean process(Transaction t)` | processes the next transaction, returning `true` if transaction was approved, `false` otherwise |

`Account` objects interact with `Transaction` objects, which have many methods including:

| Method/Constructor | Description |
|---|---|
| `public int value()` | returns the value of this transaction in pennies (could be negative, positive or zero) |

The company wishes to create a slight modification to the `Account` class that filters out zero-valued transactions. Design a new c called `FilteredAccount` whose instances can be used in place of an `Account` object but which include the extra behavior of n processing transactions with a value of 0. More specifically, the new class should indicate that a zero-valued transaction was approved but shouldn't call the `process` method in the `Account` class to process it. Your class should have a single constructor accepts a parameter of type `Client`, and it should include the following method:

| Method/Constructor | Description |
|---|---|
| `public double percentFiltered()` | returns the percent of transactions filtered out (between 0.0 and 100.0); returns 0.0 if no transactions submitted |

Assume that all transactions enter the system by a call on the `process` method described above.

**Type your solution here:**

```java
public class FilteredAccount extends Account{
    private int numFilter;
    private int transactions;

    public FilteredAccount(Client c){
        super(c);
        numFilter = 0;
        transactions = 0;
    }

    public boolean process(Transaction t){
        transactions += 1;

        if (t.value() == 0){
            numFilter += 1;
            return true;
        }

        else
            return super.process(t);
    }

    public double percentFiltered(){
        if (transactions == 0) {
            return 0.0;
        }
        return (numFilter * 100.0 / transactions);
    }
}
```

This is an **inheritance problem.** Write a Java class using inheritance. (You do not need to write any `import` statements.)

4

Submit

☑ Sound F
☑ Highligh

**You passed 1 of 1 tests.**

**test #1:** `percentFiltered()`