Code Package

CPSC 210 - Practice Final Exam

YOU MUST HAND THIS BOOKLET IN AT THE END OF THE MIDTERM BUT NOTHING YOU WRITE IN THIS BOOKLET WILL BE MARKED.

```java
public interface Wearable {
   String getName();
   double getPrice();
}
```

```java
public abstract class Clothing implements Wearable {

   public enum ClothingSize {XS, S, M, L, XL, XXL, UNI_SIZE}

   protected ClothingSize size;
   protected String name;
   protected double price;

   public Clothing(String name, ClothingSize size, double price) {
       this.name = name;
       this.size = size;
       this.price = price;
   }

   public String getName() {
       return name;
   }

   public ClothingSize getSize() {
       return size;
   }

   public double getPrice() {
       return price;
   }
}
```

```java
public class Hat extends Clothing {
   private Scarf matchingScarf;

   public Hat(String name, double price) {
       super(name, ClothingSize.UNI_SIZE, price);
       matchingScarf = null;
   }

   // REQUIRES: this hat does not have a matching scarf
   // MODIFIES: this
   // EFFECTS: adds a matching scarf for this hat
   public void addMatchingScarf(Scarf f) {
       // implementation omitted
   }

   // MODIFIES: this
   // EFFECTS: removes the matching scarf for this hat
   public void removeMatchingScarf() {
       // implementation omitted
   }

   public boolean hasMatchingScarf() {
       return matchingScarf != null;
   }
}
```

```java
public class Scarf extends Clothing {
   private Hat matchingHat;

   public Scarf(String name, ClothingSize size, double price) {
       super(name, size, price);
       matchingHat = null;
   }

   // REQUIRES: this scarf does not have a matching hat
   // MODIFIES: this
   // EFFECTS: adds a matching hat for this scarf
   public void addMatchingHat(Hat f) {
       // implementation omitted
   }

   // MODIFIES: this
   // EFFECTS: removes the matching hat for this scarf
   public void removeMatchingHat() {
       // implementation omitted
   }

   public boolean hasMatchingHat() {
       return matchingHat != null;
   }
}
```

```java
public class Handbag implements Wearable {
   private static final double DISCOUNT = 0.15;
   private String name;
   private double price;
   private boolean onSale;

   public Handbag(String name, double price) {
       this.name = name;
       this.price = price;
       this.onSale = false;
   }

   public void putOnSale() {
       this.onSale = true;
   }

   public void removeFromSale() {
       this.onSale = false;
   }

   // EFFECTS: returns price if item is not on sale otherwise
   //          returns price discounted by DISCOUNT
   public double getPrice() {
       if (onSale)
           return price;
       else
           return price * (1 - DISCOUNT);
   }

   public String getName() {
       return name;
   }
}
```

```java
public class Store {
   private static final int MAX_ITEMS = 150;
   private List<Wearable> inventory;
   private Set<Customer> customers;
   private Map<Customer, List<Wearable>> customerBaskets; // do not include
                                            // this relationship on UML diagram
   public Store() {
       inventory = new ArrayList<>();
       customers = new HashSet<>();
       customerBaskets = new HashMap<>();
   }

   // MODIFIES: this
   // EFFECTS: adds a wearable to the wearable inventory
   public void addWearable(Wearable w) {
       if (inventory.size() < MAX_ITEMS)
           inventory.add(w);
   }

   // MODIFIES: this
   // EFFECTS: adds a new customer
   public void addCustomer(Customer c) {
       customers.add(c);
   }

   // MODIFIES: this
   // EFFECTS: adds wearable w to the shopping basket of customer c
   public void addToShoppingBasket(Customer c, Wearable w) {
       // implementation omitted
   }

   // MODIFIES: this
   // EFFECTS: removes wearable w from the shopping basket of customer c
   public void removeFromShoppingBasket(Customer c, Wearable w) {
       // implementation omitted
   }

   public List<Wearable> getInventory() {
       return inventory;
   }
```

checkout method on the next page.

```java
    // MODIFIES: this
    // EFFECTS: omitted
    public void checkout(Customer c) {
        List<Wearable> basket = customerBaskets.get(c);
        double totalPrice = 0.0;
        for (Wearable w : basket)
            totalPrice += w.getPrice();

        boolean madePayment = c.makePayment(totalPrice);

        if (madePayment) {
            System.out.println("Payment received!");
            System.out.println("Shipping to: " + c.getName());

            for (Wearable w : basket) {
                inventory.remove(w);
            }

            basket.clear();
            customerBaskets.remove(c);
        } else {
            System.out.println("Payment failed!");
        }
    }
}
```

The `checkout` method is part of the `Store` class.

```java
public class Billing {
   private String creditCardNumber;
   private String billingAddress;
   private String creditCardExpiry;
   private int creditCardCVD;      // Card Verification Digits

   public void setCreditCardNumber(String number) {
       creditCardNumber = number;
   }

   public void setBillingAddress(String address) {
       billingAddress = address;
   }

   public void setCreditCardExpiry(String date) {
       creditCardExpiry = date;
   }

   public void setCreditCardCVD(int cvd) {
       creditCardCVD = cvd;
   }

   // REQUIRES: positive amount
   // EFFECTS: returns true if transaction was successful, false otherwise
   public boolean chargeCreditCard(double amount) {
       // Implementation is omitted.
       return true;
   }

   // REQUIRES: positive amount
   // EFFECTS: returns true if transaction was successful, false otherwise
   public boolean refundCreditCard(double amount) {
       // Implementation is omitted.
       return true;
   }
}
```

```
public class Customer {
   private Billing billing;
   private String name;
   private String shippingAddress;
   private String emailAddress;
   private int royaltyPoints;

   public Customer(String name, String emailAddress) {
       this.name = name;
       this.emailAddress = emailAddress;
       billing = new Billing();
       royaltyPoints = 0;
   }

   public Billing getBillingInfo() {
       return billing;
   }

   public void setBillingInfo(Billing billing) {
       this.billing = billing;
   }

   public String getName() {
       return name;
   }

   public String getShippingAddress() {
       return shippingAddress;
   }

   public String getEmailAddress() {
       return emailAddress;
   }

   public void setShippingAddress(String address) {
       shippingAddress = address;
   }

   public void setEmailAddress(String email) {
       emailAddress = email;
   }
```

See the next page for other methods in the `Customer` class.

```java
    // REQUIRES: amount > 0
    // EFFECTS: returns 10 points
    //          plus two points for every dollar spent
    protected int calcRoyaltyPoints(double amount) {
        return 10 + 2 * (int) amount;
    }

    // REQUIRES: positive amount
    // MODIFIES: this
    // EFFECTS: returns true if transaction was successful, false otherwise
    public boolean makePayment(double amount) {
        boolean madePayment = billing.chargeCreditCard(amount);
        if (madePayment) {
            royaltyPoints += calcRoyaltyPoints(amount);
            return true;
        } else {
            return false;
        }
    }

    // REQUIRES: 0 < amount <= 1000
    // EFFECTS: returns true if transaction was successful, false otherwise
    public boolean makeRefund(double amount) {
        return billing.refundCreditCard(amount);
    }
}
```

These methods belong to the `Customer` class.

```java
public class BronzeCustomer extends Customer {
   public BronzeCustomer(String name, String emailAddress) {
       super(name, emailAddress);
   }

   // REQUIRES: amount > 0
   // EFFECTS: returns 20 points
   //          plus a point for every dollar spent
   @Override
   protected int calcRoyaltyPoints(double amount) {
       return 20 + (int) amount;
   }

   // REQUIRES: 0 < amount <= 500
   // EFFECTS: returns true if transaction was successful, false otherwise
   @Override
   public boolean makeRefund(double amount) {
       return super.makeRefund(amount);
   }
}
```

```
public class GoldCustomer extends Customer {
   public GoldCustomer(String name, String emailAddress) {
       super(name, emailAddress);
   }

   // REQUIRES: amount > 0
   // EFFECTS: returns 5 points
   //          plus three points for every dollar spent
   @Override
   protected int calcRoyaltyPoints(double amount) {
       return 5 + 3 * (int) amount;
   }

   // REQUIRES: 0 < amount <= 5000
   // EFFECTS: returns true if transaction was successful, false otherwise
   @Override
   public boolean makeRefund(double amount) {
       return super.makeRefund(amount);
   }
}
```

```
public class Main {
   public static void main(String[] args) {
       Store store = new Store();

       Customer ali = new Customer("Ali", "madooei@cs.ubc.ca");
       Customer paul = new Customer("Paul", "pcarter@cs.ubc.ca");
       Customer elisa = new Customer("Elisa", "elisab@cs.ubc.ca");

       store.addCustomer(ali);
       store.addCustomer(paul);
       store.addCustomer(elisa);

       Scarf greenScarf = new Scarf("Green Scarf", ClothingSize.L, 34);
       Hat greenHat = new Hat("Green Hat", 50);
       Scarf blueScarf = new Scarf("Blue Scarf", ClothingSize.M, 30);
       Handbag rainbowBag = new Handbag("Rainbow Handbag", 235.99);
       greenScarf.addMatchingHat(greenHat);

       store.addWearable(greenScarf);
       store.addWearable(greenHat);
       store.addWearable(blueScarf);
       store.addWearable(rainbowBag);

       for (Wearable w : store.getInventory())
           System.out.println(w.getName() + "($" + w.getPrice() + ")");

       store.addToShoppingBasket(elisa, blueScarf);
       store.checkout(elisa);

       for (Wearable w : store.getInventory())
           System.out.println(w.getName() + "($" + w.getPrice() + ")");
   }
}
```