Vinjett 6 - Party-admin

Denna vinjett är skapad av en gammal IT-basgrupp och är frivillig.

Tankspridde Tom och Energiske Erik har gått med i festeriet ELiTHen. Äldre studenter varnade dem för att de skulle missa sina tentor pga hårt arbete och ändlöst festande, men Tom och Erik är smartare än så.

– Det borde gå att kombinera festeriarbete med studierna i programmering, sa Erik. – Ja, vad smart! sa Tom.

De båda mindes hur gamla avdankade festerister beklagat sig över strul med biljetter, anmälningar, sponsorer och bordsplacering vid sittningar.

– Vi borde kunna göra ett program som löser det, tyckte Tom.

Så föddes idén om PartyAdmin, ett program för hantering av gäster på studentfester.

Då du känner Tom och Erik har du lovat att hjälpa till lite med programmeringen. Projektet innehåller redan klassen Party som representerar en fest, den abstrakta klassen Person med subklasserna Student och SponsorPerson som representerar olika typer av gäster, klassen Sponsor som representerar ett företag eller en organisation som sponsrar festen, samt klassen Money som representerar svenska pengar. Källkoden i sin nuvarande status finns på sidan två och framåt. På nästa sida hittar du de sex uppgifter som har hamnat på din todo-lista.

Uppgifter

(a)

Metoden Party.pay() har ingen kropp än. Implementera kroppen med lämplig felhantering. Vid fel ska undantag kastas.

(b)

Överlagra metoden Party.addSponsor() så att den tar emot en String och en double som parametrar. Kedja den med den befintliga Party.addSponsor().

(c)

Studenter bör kunna anmäla varsin extern gäst, t.ex. en kompis hemifrån. Inför en ny klass GuestToStudent. Varje sådant objekt ska hålla reda på vilket Studentobjekt som är dess värd. Modifiera övriga klasser så att varje student kan ha en extern gäst, och att alla objekt av typen GuestToStudent hanteras som de andra gästerna.

(d)

Som du ser har programmet inget stöd för bordsplacering än. Ett första steg är att lagra uppgift om kön på gästerna, om man nu vill placera dem varannan tjej, varannan kille. Implementera en uppräkning (enum) Sex och inför den i systemet så att en placeringsmetod sedermera kan placera alla gäster efter kön.

(e)

Metoden Party.print() har ingen kropp än. Implementera kroppen så att metoden skriver ut festinformation enligt följande exempel: ELiTH-phesten i samarbete med Karlström Consulting AB och Programmeringsspecialisten AB Pris 90 kr per gäst, max 1000 gäster Anmälda gäster: Erik Karlström (är VIP), Joakim von Anka (har betalat), Pierre Anderberg (har betalat), Simon Arvidsson (har betalat), John Wilander (har betalat), Erik Janols (har inte betalat än), Eva Ragnemalm (har betalat), Jonas Wallgren (har betalat).

Om du vet att det finns en klass eller metod i Javas API som löser en viss sak men du inte kommer ihåg exakt vad den heter eller i vilket paket den finns så är det OK att införa ett eget metodanrop som löser samma sak så länge du beskriver det. Det är dock inte OK att anta att API:et innehåller fiktiva metoder som förenklar uppgiften i stil med metodenSomFixarTillLite().

(f)

Gästerna och uppgifter om huruvida de har betalat eller ej lagras i en viss typ av mappning (map). Har de klasser som lagras där ett lämpligt stöd för den mappningen, eller har Tom och Erik missat något? Motivera ditt svar och hänvisa till koden om du vill peka på något särskilt.

Klasser

```
class Party {
    private String partyName;
    private int maxGuests;
    private Money entranceFee;
    private Map<Person, Boolean> guests;
    private Map<Sponsor, Money> sponsors;
    Party(String partyName, int maxGuests, Money entranceFee) {
        this.partyName = partyName;
```

```
        this.maxGuests = maxGuests;
        this.entranceFee = entranceFee;
        this.guests = new HashMap<Person, Boolean>();
        this.sponsors = new HashMap<Sponsor, Money>();
    }
    void addGuest(Person guest) {
        addGuest(guest, false);
    }
    void addGuest(Person guest, boolean hasPaid) {
        guests.put(guest, hasPaid);
    }
    void pay(Person guest) {
        // Din kod här
    }
    boolean hasPaid(Person guest) throws NoSuchPersonException {
        if(guests.containsKey(guest)) {
            return guests.get(guest);
        } else {
            throw new NoSuchPersonException();
        }
    }
    void addSponsor(Sponsor sponsor, Money amount) {
        sponsors.put(sponsor, amount);
    }
    public void print() {
        // Din kod här
    }
}

abstract class Person {
    private final String name;
    Person(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    @Override
        public int hashCode() {
        return name.hashCode();
```

```
    }
}

class Student extends Person {
    Student(String name) {
        super(name);
    }
}

class SponsorPerson extends Person {
    private final Sponsor organization;
    SponsorPerson(String name, Sponsor organization) {
        super(name);
        this.organization = organization;
    }
    Sponsor getOrganization() {
        return organization;
    }
}

class Sponsor {
    private final String name;
    Sponsor(String name) {
        this.name = name;
    }
    @Override
        public String toString() {
        return name;
    }
    @Override
        public int hashCode() {
        return name.hashCode();
    }
}

class Money implements Cloneable {
    double amount;
    Money(double amount) {
        if(amount < 0.0) {
            throw new IllegalArgumentException();
```

```
        } else {
            this.amount = amount;
        }
    }
    @Override
        protected Money clone() {
        try {
            return (Money)super.clone();
        } catch(CloneNotSupportedException e) {
            throw new AssertionError(); // Should never happen
        }
    }
    @Override
        public String toString() {
        return Double.toString(amount);
    }
}
```