

7.7 開發設計

演習問題



一. ユースケース図

時間： 10 分

- 人事システムを作成する予定です。

人事システムでは、「登録担当者」というアクターが存在し、登録担当者は、「社員を登録する」、「社員を更新する」、「社員を削除する」、「社員を検索する」というユースケースを持ちます。この条件のユースケース図を描いてください。



二. ユースケース図2

時間： 10 分

- ECサイトを作成する予定です。
ECサイトでは、「ユーザー」というアクターが存在し、
皆さんは、ユーザーがECサイトを作るうえで必要なユース
ケースを記載していただきます。
どのようなユースケースがあるかをユースケース図を描い
て下さい。

三. アクティビティ図

時間： 15 分

- 以下の自販機アクティビティ図を描いてください。

- 硬貨を入れる
- 合計120
 - ・ 合計<120の時：硬貨を受け付ける
 - ・ 合計 = 120の時：飲み物を出す
 - ・ 合計>120の時：お釣りを出す
- 飲み物を出す。



四. アクティビティ図並列処理 時間： 15 分

- 以下の仕様を読みアクティビティ図を描いてください。
 - ユーザーがメールを確認する
 - 新着メールがあった場合
 - ・ 返事を書く
 - ・ スпамを削除する
 - 新着メールがない場合
 - ・ 処理を終了する

五. シーケンス図

時間： 15 分

- 以下の仕様を読みシーケンス図を描いてください。
 - オブジェクトは「社員」、「鉄道」、「飛行機」
 - 社員の出張先が国内の場合は、鉄道のチケットを注文する
 - 社員の出張先が海外の場合は、飛行機のチケットを注文する

五. シーケンス図

時間： 20 分

- 以下の仕様を読みシーケンス図を描いてください。
 - オブジェクトは「客」、「店員」、「レジ」
 - 商品がある分だけレジ打ちをする
 - レジを打った結果を店員が取得する
 - 客が店員にお金を払い、お金をレジにいれる
 - レジからレシートを店員に、店員からレシートを客に渡す
 - 店員から商品を客に渡す



七. クラス図

時間： 15 分

- 以下のソースのクラス図を作成しなさい。
- P10までソースがあります。

```
public class Animal {
    protected String name;

    public Animal(String name) {
        this.name = name;
    }

    public void eat(String food) {
        System.out.println(this.name + " ate " + food + ".");
    }
}
```

```
public interface Run {
    void run();
}
```



七. クラス図

時間： 15 分

- 以下のソースのクラス図を作成しなさい。
- P10までソースがあります。

```
public class Car implements Runnable {
    private String brand;

    public Car(String brand) {
        this.brand = brand;
    }

    @Override
    public void run() {
        System.out.println("A " + brand + " car is running! beep-- beep--");
    }

    public String getBrand() {
        return brand;
    }
}
```



七. クラス図

時間： 15 分

- 以下のソースのクラス図を作成しなさい。

```
public class Cat extends Animal implements Runnable {
    public Cat(String name) {
        super(name);
    }

    @Override
    public void eat(String food) {
        System.out.print(name + " ate " + food + ", ");
        meow(); // 猫が食べた後にニャーと鳴く
    }

    public void meow() {
        System.out.println("meow~");
    }

    @Override
    public void run() {
        System.out.print(name + " is running! ");
        meow();
    }
}
```

```
public class Dog extends Animal implements Runnable {
    public Dog(String name) {
        super(name);
    }

    @Override
    public void eat(String food) {
        System.out.print(name + " ate " + food + ", ");
        bowWow(); // 犬が食べた後に吠える
    }

    void bowWow() {
        System.out.println("BOWWOW!!!");
    }

    @Override
    public void run() {
        System.out.print(name + " is running! ");
        bowWow();
    }
}
```