Faculté des Sciences
Aix*Marseille Université

Année universitaire

Site : ☐ Luminy ☐ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☐ 1er semestre ☐ 2ème semestre ☐ Session 2 Durée de l'épreuve :
Examen de : L2 Nom du diplôme : Licence d'informatique
Code du module : SIN3U02 Libellé du module : Programmation 2
Calculatrices autorisées : NON Documents autorisés : OUI

---

1 Correction de l'examen

Vous trouverez ci-dessous une correction possible de l'examen. C'est une des corrections possibles mais pas la seule.

2 Gestion d'un restaurant

Le but de ce sujet d'examen est de créer des classes permettant de représenter des réservations et des commandes d'un restaurant.

2.1 Classe `Client`

```java
public class Client {
  public final String name;

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

  @Override
  public String toString() {
    return name;
  }
}
```

2.2 Interface `Item`

```java
public interface Item {
  String toString();
  int getPrice();
}
```

2.3 Classe `Dish`

```java
public class Dish implements Item{

  public Dish(int price, String name) {
    this.name = name;
    this.price = price;
  }

  @Override
  public int getPrice() {
    return price;
  }
```

```
  @Override
  public String toString() {
    return name + " : " + getPrice() + "€";
  }

}
```

2.4 Classe ThreeCourseMeal

```
public class ThreeCourseMeal implements Item{

  public ThreeCourseMeal(String name,
                         int price,
                         String starter,
                         String mainDish,
                         String dessert) {
    this.name = name;
    this.price = price;
    this.starter = starter;
    this.mainDish = mainDish;
    this.dessert = dessert;
  }

  @Override
  public int getPrice() {
    return price;
  }

  @Override
  public String toString() {
    return name +
            " (" + starter +
            " + " + mainDish +
            " + " + dessert +
            ")" + " : " +
            getPrice() + "€";
  }
}
```

2.5 Classe Table

```
public class Table {
  private static int totalNumberOfTables = 0;
  private final int id;
  private final int capacity;

  public Table(int capacity) {
    this.id = totalNumberOfTables++;
    this.capacity = capacity;
  }

  public int getCapacity() {
    return capacity;
  }

  @Override
  public String toString() {
```

```
    return "Table " +
            id +
            " (" +
            capacity +
            " places)";
  }
}
```

2.6 Classe Booking

```
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;

public class Booking {
  private final LocalTime arrivalTime;
  private final Client client;
  private int numberOfPeople;
  private List<Item> ordereditems;

  public Booking(Client client, int numberOfPeople, LocalTime arrivalTime) {
    this.arrivalTime = arrivalTime;
    this.client = client;
    this.numberOfPeople = numberOfPeople;
    ordereditems = new ArrayList<>();
  }

  public void addOrder(Item item){
    ordereditems.add(item);
  }
}
```

2.7 Méthode du calcul du prix

```
  public int billAmount(){
    int sum = 0;
    for (Item item : ordereditems){
      sum += item.getPrice();
    }
    return sum;
  }
```

2.8 Classe Restaurant

```
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Restaurant {
  public List<Table> tables = new ArrayList<>();
  private final Map<Table, Booking> bookedTables = new HashMap<>();

  public void addTable(int capacity) {
    tables.add(new Table(capacity));
  }
```

```
  public boolean tableIsBooked(Table table){
    return bookedTables.containsKey(table);
  }
}
```

2.9 Méthode `findAppropriateTable`

```
  private Table findAppropriateTable(int numberOfPeople){
    Table appropriateTable =  null;
    for(Table table: tables){
      if (! tableIsBooked(table)
              && table.getCapacity() >= numberOfPeople
              && (appropriateTable == null
              || appropriateTable.getCapacity() > table.getCapacity())){
        appropriateTable = table;
      }
    }
    return appropriateTable;
  }
```

2.10 Méthode `book`

```
  public Table book(Client client, int numberOfPeople, LocalTime arrivalTime){
    Table appropriateTable = findAppropriateTable(numberOfPeople);
    bookedTables.put(appropriateTable, new Booking(client, numberOfPeople, arrivalTime));
    return appropriateTable;
  }
```

2.11 Prendre une commande

Classe `Restaurant`

```
  public void addOrder(Item item, int quantity, Table table){
    bookedTables.get(table).addOrder(item, quantity);
  }

  public void addOrder(Item item, Table table){
    addOrder(item, 1, table);
  }
```

Classe `Booking`

```
  public void addOrder(Item item, int quantity){
    for(int index = 0; index < quantity; index++){
      addOrder(item);
    }
  }
```

2.12 Affichage

Classe `Restaurant`

```
  public String toString(){
    String result = "";
    for (Table table: tables) {
      result += table + "\n";
      if (tableIsBooked(table)){
        result += bookedTables.get(table) + "\n";
      }
```

```
      else{
        result += "vide\n";
      }
    }
    return result;
  }
```

Classe Booking

```
  @Override
  public String toString() {
    String result = "Réservation de " +
            client +
            " (" +
            numberOfPeople +
            ", " +
            arrivalTime +
            ")\n";
    for (Item item : ordereditems){
      result += item + "\n";
    }
    return result + "Total : " + billAmount() + "€\n";
  }
```

2.13 Factorisation du code

```
public abstract class AbstractItem implements Item {
  private final int price;
  protected final String name;

  public AbstractItem(int price, String name) {
    this.price = price;
    this.name = name;
  }

  @Override
  public int getPrice() {
    return price;
  }

  @Override
  public String toString() {
    return getDescription() + " : " + getPrice() + "€";
  }

  public abstract String getDescription();
}

public class Dish extends AbstractItem implements Item{

  public Dish(int price, String name) {
    super(price, name);
  }

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

```java
public class ThreeCourseMeal extends AbstractItem implements Item{
  private final String starter;
  private final String mainDish;
  private final String dessert;

  public ThreeCourseMeal(String name,
                         int price,
                         String starter,
                         String mainDish,
                         String dessert) {
    super(price, name);
    this.starter = starter;
    this.mainDish = mainDish;
    this.dessert = dessert;
  }

  @Override
  public String getDescription() {
    return name +
            " (" + starter +
            " + " + mainDish +
            " + " + dessert +
            ")";
  }
}
```