Programmation Objet Avancée - TD3

—o000o——o000o—

Héritage et interface

1 Téléphones et smartphones

Créez 2 classes `FeaturePhone` et `SmartPhone` ayant pour attributs communs :

- Une marque
- Un modèle
- Un numéro de téléphone
- Un opérateur téléphonique
- Un attribut spécifiant si l'appareil est bloqué chez cet opérateur

La classe `SmartPhone` possède en plus :

- Un attribut spécifiant le système d'exploitation (Android, iOS)
- Un attribut spécifiant si le smartphone possède un appareil photo
- Un attribut spécifiant si le smartphone possède un GPS

Par défaut, tous les téléphones sont bloqués chez un opérateur téléphonique.

1. Créez les classes nécessaires (attributs, constructeur, ...)
2. Créez également une méthode permettant d'afficher l'ensemble des valeurs d'un téléphone, une méthode permettant de débloquer un téléphone ainsi qu'une méthode permettant de changer le numéro de téléphone d'un téléphone
3. Créez une méthode `equals(Object o)` qui retourne `true` si et seulement si les 2 téléphones sont identiques (même modèle et de même marque)

Solution :

```
1  package poo.tp3.phone;
2
3  import java.util.Objects;
4
5  public class Phone implements Call {
6      private String brand;
7      private String model;
8      private String phoneNumber;
9      private String phoneOperator;
10     private boolean isBlocked; // true by default
11
```

```
12      public Phone(String brand, String model, String phoneNumber,
            String phoneOperator) {
13          this.brand = brand;
14          this.model = model;
15          this.phoneNumber = phoneNumber;
16          this.phoneOperator = phoneOperator;
17      }
18
19      public String getBrand() {
20          return brand;
21      }
22
23      public void setBrand(String brand) {
24          this.brand = brand;
25      }
26
27      public String getModel() {
28          return model;
29      }
30
31      public void setModel(String model) {
32          this.model = model;
33      }
34
35      public String getPhoneNumber() {
36          return phoneNumber;
37      }
38
39      public void setPhoneNumber(String phoneNumber) {
40          this.phoneNumber = phoneNumber;
41      }
42
43      public String getPhoneOperator() {
44          return phoneOperator;
45      }
46
47      public void setPhoneOperator(String phoneOperator) {
48          this.phoneOperator = phoneOperator;
49      }
50
51      public boolean isBlocked() {
52          return isBlocked;
53      }
54
55      public void setBlocked(boolean blocked) {
56          isBlocked = blocked;
57      }
58
59      public void unLock() {
```

```
60          isBlocked = false;
61      }
62
63      @Override
64      public void call(Phone phone) {
65          System.out.println(this.phoneNumber + " call " + phone.
                phoneNumber);
66      }
67
68      @Override
69      public boolean equals(Object o) {
70          if (this == o) return true;
71          if (o == null || getClass() != o.getClass()) return false;
72          Phone phone = (Phone) o;
73          return Objects.equals(brand, phone.brand) &&
74              Objects.equals(model, phone.model);
75      }
76
77      @Override
78      public String toString() {
79          return "Phone{" +
80              "brand='" + brand + '\'' +
81              ", model='" + model + '\'' +
82              ", phoneNumber='" + phoneNumber + '\'' +
83              ", phoneOperator='" + phoneOperator + '\'' +
84              ", isBlocked=" + isBlocked +
85              '}';
86      }
87  }
```

java/Phone.java

Solution :

```
1  package poo.tp3.phone;
2
3  public class FeaturePhone extends Phone {
4      public FeaturePhone(String brand, String model, String phoneNumber
           , String phoneOperator) {
5          super(brand, model, phoneNumber, phoneOperator);
6      }
7  }
```

java/FeaturePhone.java

Solution :

```java
package poo.tp3.phone;

public class SmartPhone extends Phone {
    private String os;
    private boolean isAPN;
    private boolean isGPS;

    public SmartPhone(String brand, String model, String phoneNumber,
       String phoneOperator, String os, boolean isAPN, boolean isGPS)
       {
        super(brand, model, phoneNumber, phoneOperator);
        this.os = os;
        this.isAPN = isAPN;
        this.isGPS = isGPS;
    }

    public String getOs() {
        return os;
    }

    public void setOs(String os) {
        this.os = os;
    }

    public boolean isAPN() {
        return isAPN;
    }

    public void setAPN(boolean APN) {
        isAPN = APN;
    }

    public boolean isGPS() {
        return isGPS;
    }

    public void setGPS(boolean GPS) {
        isGPS = GPS;
    }

    @Override
    public String toString() {
        return "SmartPhone{" +
            "os='" + os + '\'' +
            ", isAPN=" + isAPN +
            ", isGPS=" + isGPS +
            "} " + super.toString();
    }
```

```
}
```

java/SmartPhone.java

4. Dans une classe de test, créez 2 `FeaturePhone` et 2 `SmartPhone`
5. Ajoutez ces 4 téléphones nouvellement créés dans la structure de données de votre choix
6. Appliquer la méthode de déblocage à l'ensemble des téléphones
7. Appliquer la méthode d'affichage à l'ensemble des téléphones
8. ⋆ ⋆ ⋆ **Pour aller plus loin**, utilisez la méthode `forEach` présent dans Java 8 pour répondre aux questions 6 et 7

Solution :

```java
package poo.tp3.phone;

import java.util.ArrayList;
import java.util.List;

public class PhoneTestCase {
    public static void main(String[] args) {
        List<Phone> phones = new ArrayList<>();

        FeaturePhone featurePhone1 = new FeaturePhone("Nokia", "3310",
            "0678766567", "Free");
        FeaturePhone featurePhone2 = new FeaturePhone("Nokia", "3311",
            "0678766555", "Orange");

        SmartPhone smartPhone1 = new SmartPhone("Samsung", "Galaxy S20
           ", "0678733567", "SFR", "Android", true, true);
        SmartPhone smartPhone2 = new SmartPhone("Apple", "iPhone 11",
           "0698733567", "Sosh", "iOS", true, true);

        phones.add(featurePhone1);
        phones.add(featurePhone2);
        phones.add(smartPhone1);
        phones.add(smartPhone2);

        for (Phone phone : phones) {
            phone.unLock();
        }

        for (Phone phone : phones) {
            System.out.println(phone);
        }

        // Java 8
```

```
30          phones.forEach(Phone::unLock);
31          phones.forEach(System.out::println);
32
33          smartPhone1.call(smartPhone2);
34      }
35  }
```

java/PhoneTestCase.java

9. Créez une interface `Call` contenant une unique méthode `call(Phone phone)` permettant d'afficher les 2 numéros en communication

Solution :

```
1  package poo.tp3.phone;
2
3  public interface Call {
4      void call(Phone phone);
5  }
```

java/Call.java

2 Pages et enveloppe

Nous souhaitons modéliser un courrier. Pour ceci, nous allons définir les classes `Page` et `Enveloppe`.

Une page est composée :

- D'un recto (une chaîne de caractères)
- Et d'un verso (une autre chaîne de caractères)

Un seul côté est visible à la fois.

1. Écrire la classe `Page` contenant les méthodes suivantes :
   - Un constructeur prenant en paramètre 2 chaînes de caractères représentant le recto et le verso d'une feuille. Initialement, seul le recto de la feuille est visible
   - La redéfinition de la méthode `String toString()`. Pour rappel, un seul côté est visible
   - Une méthode `void flip()` permettant de retourner la page (rend visible le côté qui ne l'était pas)

Solution :

```
1  package poo.tp3.mail;
2
3  public class Page {
4      private String recto;
5      private String verso;
```

```java
    private boolean isReadableRecto = true;

    public Page(String recto, String verso) {
        this.recto = recto;
        this.verso = verso;
    }

    public String getRecto() {
        return recto;
    }

    public void setRecto(String recto) {
        this.recto = recto;
    }

    public String getVerso() {
        return verso;
    }

    public void setVerso(String verso) {
        this.verso = verso;
    }

    public boolean isReadableRecto() {
        return isReadableRecto;
    }

    public void setReadableRecto(boolean readableRecto) {
        isReadableRecto = readableRecto;
    }

    public void flip() {
        isReadableRecto = !isReadableRecto;
    }

    private String getReadablePage() {
        return isReadableRecto ? recto : verso;
    }

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

java/Page.java

Solution :

```java
package poo.tp3.mail;

public enum PageFactory {
    INSTANCE;

    int maxPage;
    boolean isInit = false;
    private int numInstance;

    public void greenThreshold(int threshold) {
        if (threshold <= 0) {
            throw new IllegalArgumentException("threshold must be
                positive");
        }

        this.maxPage = threshold;
        isInit = true;
    }

    public Page getPage(String recto, String verso) {
        if (!isInit) {
            throw new AssertionError("Please, initialized me !");
        }

        if (numInstance >= maxPage) {
            throw new AssertionError("Stop waste!");
        }

        numInstance++;

        return new Page(recto, verso);
    }
}
```

java/PageFactory.java

Solution :

```java
package poo.tp3.mail;

public class PageFactoryTestCase {
    public static void main(String[] args) {
        PageFactory factory = PageFactory.INSTANCE;
        factory.greenThreshold(3);
        factory.getPage("", "");
        factory.getPage("", "");
        factory.getPage("", "");
        factory.getPage("", "");
```

```
    }
}
```

java/PageFactoryTestCase.java

2. Testez la classe `Page`
3. Écrire la classe `Enveloppe`. Elle aura pour propriétés :
   - Un tableau de `Page`
   - Le nombre maximal de pages qu'une enveloppe peut contenir
   - L'état de l'enveloppe (ouverte ou fermée)

   Et pour méthodes :
   - `void open()` et `void close()` permettant respectivement d'ouvrir ou de fermer l'enveloppe
   - `boolean addPage(Page page)` permettant d'ajouter une page à une enveloppe. N'oubliez pas de contrôler le nombre maximal de pages autorisées
   - ⋆⋆⋆ **Pour aller plus loin**, implémentez votre propre `ArrayList` et redéfinissez la méthode `boolean add(Page page)`
   - `void read()` permettant de lire l'ensemble des pages contenues dans l'enveloppe (recto et verso). La lecture n'est possible que si l'enveloppe est ouverte
   - `void size()` permettant de retourner le nombre de pages contenues dans une enveloppe
4. Testez la classe `Enveloppe`
5. ⋆⋆ Le papier étant consommateur d'arbres, nous essayons de limiter au maximum son utilisation. Modifiez la classe `Page` en y ajoutant, notamment, la méthode `void greenThreshold(int threshold)`, qui fixe le nombre maximum de pages au-delà duquel, le message "**Halte au gaspillage !**" sera affiché avant chaque création d'une nouvelle page.

Solution :

```java
package poo.tp3.mail;

import java.util.ArrayList;
import java.util.List;

public class Envelope {
    private static final int DEFAULT_MAX_PAGES = 2;
    private final List<Page> pages = new ArrayList<>(DEFAULT_MAX_PAGES);
    private boolean isOpen; // true by default

    public void open() {
        isOpen = true;
    }

    public void close() {
        isOpen = false;
    }
```

```
18
19      public boolean addPage(Page page) {
20          if (this.size() < DEFAULT_MAX_PAGES) {
21              return pages.add(page);
22          }
23
24          System.out.println("Number of pages reached");
25          return false;
26      }
27
28      public void read() {
29          if (isOpen) {
30              if (pages.isEmpty()) {
31                  System.out.println("This envelope doesn't contain pages");
32              } else {
33                  for (Page page : pages) {
34                      System.out.println(page);
35                      page.flip();
36                      System.out.println(page);
37                  }
38              }
39          } else {
40              System.out.println("This envelope isn't open");
41          }
42      }
43
44      public int size() {
45          return pages.size();
46      }
47  }
```

java/Envelope.java

Solution :

```
1   package poo.tp3.mail;
2
3   public class EnvelopeTestCase {
4       public static void main(String[] args) {
5           Page page1 = new Page("recto page1", "verso page1");
6           Page page2 = new Page("recto page2", "verso page2");
7           Page page3 = new Page("recto page3", "verso page3");
8           //System.out.println(page1);
9           //page1.flip();
10          //System.out.println(page1);
11
12          Envelope envelope = new Envelope();
13          envelope.addPage(page1);
14          envelope.addPage(page2);
```

```
15          envelope.addPage(page3);
16          envelope.read();
17          envelope.close();
18          envelope.read();
19      }
20  }
```

java/EnvelopeTestCase.java