Espace Malraux - Place Saint Clair
BP 86
14203 Hérouville Saint-Clair Cedex

le cnam Normandie

NFP121 : Programmation avancée

Enseignant : Philippe BRUTUS

Année universitaire 2024-2025

Sujet d'examen de première session

Date : vendredi 20 juin 2025

Horaires : 13h40 - 16h10 (durée : 2h30)

Préférences d'impression : recto, couleur

Modalités pratiques

Éléments :	**autorisés**	**non autorisés**
Supports de cours (polycopiés)		x
Documents manuscrits (prise de notes)	Recto-verso A4	
Calculatrice		x

Autres consignes / remarques :

- Tout appareil communicant ou de stockage de données (téléphone, tablette, ordinateur …) doit être éteint et rangé.
- Lisez attentivement le sujet.
- Si vous faîtes une hypothèse par rapport au sujet, indiquez-le explicitement.

Barème de notation :

1) Questions de cours : 5 points
   a) 1 point
   b) 2 points
   c) 2 points
2) Singleton : 1,5 point
   a) 0,5 point
   b) 1 point
3) Commande : 9,5 points
   a) 1 point
   b) 3 points
   c) 2 points
   d) 3,5 points
4) Exécution différée : 4 points

1) Questions de cours

a) **Expliquer en quelques phrases ce qu'est un patron de conception.**

b) **Citer le nom de chacune des 3 familles de patrons de conception et expliquer à quoi servent les patrons de cette famille.**

c) **Citer le nom d'un patron dans chacune des 3 familles et expliquer son utilité.**

Paiement en plusieurs fois chez un commerçant

Pour ses clients commerçants qui ont un compte bancaire professionnel, une banque met à disposition un terminal de paiement électronique. Ce dernier permet aux clients du commerçant de payer leurs achats par carte bancaire…

- en une fois, le jour de l'achat ;
- en une fois, un mois après l'achat ;
- en plusieurs fois à parts égales, le jour de l'achat puis une fois par mois.

On définit à cet effet une classe Banque proposant des services de gestion des comptes :

- nouveauCompte qui crée un nouveau compte bancaire et retourne son numéro ;
- compte qui retourne le compte bancaire dont le numéro est fourni ;

et des services de paiement par carte bancaire :

- paiement qui effectue un paiement en débitant du montant fourni en contimes le compte attaché à la carte bancaire dont le numéro est fourni et crédite du même montant le compte dont le numéro est fourni ;
- paiementNFois qui effectue un premier paiement en débitant le compte attaché à la carte bancaire dont le numéro est fourni et crédite le compte dont le numéro est fourni puis enregistre des paiements programmés espacés d'un mois pour les effectuer le jour venu.

Voici le code de la classe CompteBancaire :

```
public class CompteBancaire {
    private int numéro;
    private int solde;
    public CompteBancaire(int numéroDeCompte) {
        numéro = numéroDeCompte;
        solde = 0;
    }
    public void dépose(int montant) throws Exception {
        if (montant > 0) solde += montant;
        else throw new Exception("Dépôt d'un montant négatif impossible !");
    }
    public void retire(int montant) throws Exception {
        if (montant > 0) {
            if (solde >= montant) solde -= montant;
            else throw new Exception("La banque n'autorise pas de découvert !");
        } else throw new Exception("Retrait d'un montant négatif impossible !");
    }
    public int solde() {
        return solde;
    }
    public int numero() {
        return numéro;
    }
    public String toString() {
        return "numero : " + numéro + " solde : " + solde;
    }
}
```

On notera que le solde et les montants déposés ou retirés sont exprimés sous forme d'entiers pour éviter les problèmes d'approximation des nombres à virgule. Ils sont donc exprimés en contimes.

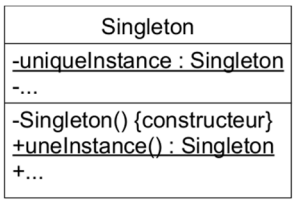
Voici le code partiel de la classe Banque :

```
public class Banque {
  private static int nombreDeComptes = 0;
  private List<CompteBancaire> comptesGérés;
  private List<CommandeProgrammée> paiementsProgrammés;
  public Banque() {
     comptesGérés = new LinkedList<CompteBancaire>();
     paiementsProgrammés = new LinkedList<CommandeProgrammée>();
  }
  public int nouveauCompte() {
     nombreDeComptes++;
     int numéroDuNouveauCompte = nombreDeComptes;
     comptesGérés.add(new CompteBancaire(numéroDuNouveauCompte));
     return numéroDuNouveauCompte;
  }
  public CompteBancaire compte(int numéro) {
     for(CompteBancaire cpt : comptesGérés) {
        if (cpt.numero() == numéro) return cpt;
     }
     return null;
  }
  public void affichePaiementsProgrammés() {
     if (paiementsProgrammés.size() > 0)
        System.out.println("Paiements programmés...");
     for(CommandeProgrammée c : paiementsProgrammés) {
        System.out.println(c);
     }
  }
  public void paiement(int numCptBénéficiaire, String numéroCBPayeur, int montant)
  throws Exception {
     System.out.print("Le " + Date.aujourdhui());
     System.out.println(" transaction avec la banque du payeur...");
     System.out.print("  pour paiement de ") ;
     System.out.println(String.format("%.2f", (float)montant/(float)100));
     System.out.println("  sur le compte " + numCptBénéficiaire);
     System.out.println("  depuis le compte du porteur de la carte " + numéroCBPayeur);
     CompteBancaire cpt = compte(numCptBénéficiaire);
     cpt.dépose(montant);
  }
  public void paiementDifféré(int numCptBénéficiaire, String numCBClient,
                              int montant, Date quand) {
     // à compléter...
  }
  public void paiementNFois(int numCptBénéficiaire, String numCBClient,
                            int montant, int nombre) throws Exception {
     // à compléter...
  }
  public void traitePaiementsProgrammésDuJour() throws Exception {
     // à compléter...
  }
}
```

2) Singleton

L'application de gestion des comptes et des paiements par carte bancaire n'utilise qu'une seule instance de la classe banque.

Pour que la classe Banque n'ait qu'une seule instance, on utilise le patron de conception singleton. Pour empêcher d'instancier la classe plus d'une fois, le constructeur d'un singleton est privé. Une méthode publique et de classe permet d'obtenir une instance en la créant si elle n'existe pas ou en renvoyant l'instance déjà créée dans le cas contraire :

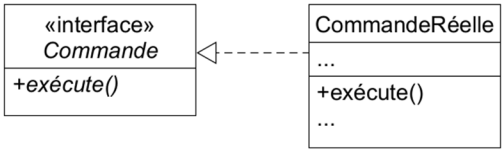


a) **Donner la déclaration de la variable représentant cette banque.**

b) **Donner le code de la méthode laBanque() qui renvoie cette banque en l'ayant préalablement créée si elle ne l'était pas.**

3) Commande

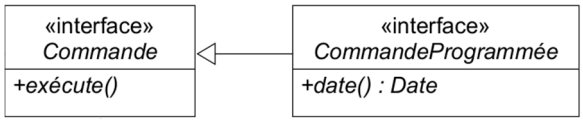
Pour gérer les paiements différés, on utilise le patron de conception commande qui consiste à représenter une action par un objet plutôt que de l'exécuter.



Pour exécuter l'action, on utilise l'opération sans paramètre `exécute()`, proposée par tout objet de type Commande.

Bien sûr, une commande réelle peut avoir des paramètres et l'objet qui la représente les définit comme des variables d'instance, initialisées à la construction et utilisées dans le corps de la méthode qui définit l'opération `exécute()`.

Comme on souhaite que l'action soit exécutée à une date précise, on définit un type particulier de commande, la commande programmée. Celle-ci, en plus de proposer une opération `exécute()` pour réaliser l'action, propose une opération `date()` qui renvoie la date à laquelle l'action est prévue.



Voici le code de la classe Date :

```
public class Date {
   private byte jour, mois;
   private int année;
   private static Date aujourdhui = new Date(20, 6, 2025);
   public Date(int j, int m, int a) {
      jour = (byte) j;
      mois = (byte) m;
      année = a;
   }
   public static Date aujourdhui() {
      return aujourdhui;
   }
```

```
    public Date copie() {
       return new Date(jour, mois, année);
    }
    public boolean egale(Date autre) {
       return année == autre.année && mois == autre.mois && jour == autre.jour;
    }
    public static byte nombreDeJoursDuMois(int mois, int annee) {
       switch (mois) {
       case 1:
       case 3:
       case 5:
       case 7:
       case 8:
       case 10:
       case 12: return 31;
       case 4:
       case 6:
       case 9:
       case 11: return 30;
       case 2: if (bissextile(annee)) return 29;
               return 28;
       default: return 0;
       }
    }
    public static boolean bissextile(int annee) {
       if (annee % 4 != 0) return false;
       if (annee % 100 == 0) return annee % 400 == 0;
       return true;
    }
    public void plusUnJour() {
       jour++;
       if (jour > nombreDeJoursDuMois(mois, année)) {
          jour = 1;
          plusUnMois();
       }
    }
    public void plusUnMois() {
       mois++;
       if (mois > 12) {
          mois = 1;
          année++;
       } else {
          if (mois == 2) {
             byte max = Date.nombreDeJoursDuMois(mois, année);
             if (jour > max)
                jour = max;
          }
       }
    }
    public String toString() {
       return String.format("%02d/%02d/%04d", jour, mois, année);
    }
}
```

a) **Donner la définition de l'interface CommandeProgrammée.**

b) **Donner la définition de la classe CommandePaiementCB qui est une commande programmée permettant l'exécution différée d'un paiement par carte bancaire.**
L'exécution de cette commande de paiement par carte bancaire consiste donc à appeler la méthode paiement de la classe Banque.

c) **Donner la définition de la méthode paiementDifféré de la classe Banque.**
Cette méthode crée une commande programmée de paiement par carte bancaire et l'ajoute aux paiements programmés.

d) **Donner la définition de la méthode paiementNFois de la classe Banque.**
Cette méthode calcule le montant du premier paiement et exécute de premier paiement. Elle calcule le montant et la date de chacun des autres paiements et les enregistre comme des commandes programmées.
Attention, le montant du premier paiement n'est pas toujours le même que celui des autres paiements car le montant total n'est pas toujours divisible par le nombre de paiements. Par exemple 14€ en 3 fois, c'est-à-dire 1400 cents en 3 fois, donnent 466 cents + 467 cents + 467 cents).

4) Exécution différée des commandes enregistrées

Chaque jour, la banque recherche les paiements programmés pour cette date. Pour chacun d'eux, elle réalise le paiement et retire la commande programmée des commandes enregistrées.

Donner la définition de la méthode traitrePaiementsProgrammés de la classe Banque.

Annexe 1 : Exemple de programme mettant en œuvre des paiements en plusieurs fois

Voici le code d'un programme d'exemple :

```
public class EssaiPNF {
   private static Banque banque = Banque.laBanque();
   public static void paiement1Fois( int numéroCompte,
                                     String numéroCB,
                                     float montant)
   throws Exception {
      System.out.println("Paiement en 1 fois de " + String.format("%.2f", montant));
      banque.paiement(numéroCompte, numéroCB, (int)(100 * montant));
   }
   public static void paiementNFois( int numéroCompte,
                                     String numéroCB,
                                     float montant,
                                     int nFois)
   throws Exception {
      System.out.print("Paiement en " + nFois  + " fois de ");
      System.out.println(String.format("%.2f", montant));
      banque.paiementNFois(numéroCompte, numéroCB, (int)(100 * montant), nFois);
   }
   public static void main(String[] args) throws Exception {
      int numéroCompteBénéficiare = banque.nouveauCompte();
      paiement1Fois(numéroCompteBénéficiare, "456723", 11.11f);
      paiementNFois(numéroCompteBénéficiare, "345678", 222.22f, 3);
      paiementNFois(numéroCompteBénéficiare, "234567", 3333.33f, 10);
      for(int i = 1; i <= 10; i++) {
         banque.affichePaiementsProgrammés();
         System.out.println("----------------------" + i + " mois après ---------------");
         Date.aujourdhui().plusUnMois();
         banque.traitePaiementsProgrammésDuJour();
      }
   }
}
```

... et sa trace d'exécution :

```
Paiement en 1 fois de 11,11
Le 20/06/2025 transaction avec la banque du payeur...
  pour paiement de 11,11
  sur le compte 1
  depuis le compte du porteur de la carte 456723
Paiement en 3 fois de 222,22
Le 20/06/2025 transaction avec la banque du payeur...
  pour paiement de 74,08
  sur le compte 1
  depuis le compte du porteur de la carte 345678
Le 20/06/2025, programmation d'un paiement de 74,07 le 20/07/2025
Le 20/06/2025, programmation d'un paiement de 74,07 le 20/08/2025

Paiement en 10 fois de 3333,33
Le 20/06/2025 transaction avec la banque du payeur...
  pour paiement de 333,36
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/07/2025
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/08/2025
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/09/2025
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/10/2025
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/11/2025
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/12/2025
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/01/2026
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/02/2026
Le 20/06/2025, programmation d'un paiement de 333,33 le 20/03/2026
```

```
Paiements programmés...
Le 20/07/2025 paiement de 74,07 sur le compte 1 depuis le compte du porteur de la carte 345678
Le 20/08/2025 paiement de 74,07 sur le compte 1 depuis le compte du porteur de la carte 345678
Le 20/07/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/08/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/09/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/10/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/11/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/12/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------1 mois après ---------------
Le 20/07/2025 transaction avec la banque du payeur...
  pour paiement de 74,07
  sur le compte 1
  depuis le compte du porteur de la carte 345678
Le 20/07/2025 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/08/2025 paiement de 74,07 sur le compte 1 depuis le compte du porteur de la carte 345678
Le 20/08/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/09/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/10/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/11/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/12/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------2 mois après ---------------
Le 20/08/2025 transaction avec la banque du payeur...
  pour paiement de 74,07
  sur le compte 1
  depuis le compte du porteur de la carte 345678
Le 20/08/2025 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/09/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/10/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/11/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/12/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------3 mois après ---------------
Le 20/09/2025 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/10/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/11/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/12/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------4 mois après ---------------
Le 20/10/2025 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/11/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/12/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
```

```
----------------------5 mois après ----------------
Le 20/11/2025 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/12/2025 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------6 mois après ----------------
Le 20/12/2025 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/01/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------7 mois après ----------------
Le 20/01/2026 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/02/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------8 mois après ----------------
Le 20/02/2026 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
Paiements programmés...
Le 20/03/2026 paiement de 333,33 sur le compte 1 depuis le compte du porteur de la carte 234567
----------------------9 mois après ----------------
Le 20/03/2026 transaction avec la banque du payeur...
  pour paiement de 333,33
  sur le compte 1
  depuis le compte du porteur de la carte 234567
----------------------10 mois après ----------------
```

Annexe 2 : Quelques API de java

Interface Collection<E>
All Superinterfaces:
Iterable<E>
All Known Subinterfaces:
..., List<E>, …
All Known Implementing Classes:
…, ArrayList, …, LinkedList, …, Vector

```
public interface Collection<E>
extends Iterable<E>
```

Method Summary	
boolean	**add**(E e) Ensures that this collection contains the specified element (optional operation).
void	**clear**() Removes all of the elements from this collection (optional operation).
boolean	**contains**(Object o) Returns true if this collection contains the specified element.
boolean	**equals**(Object o) Compares the specified object with this collection for equality.
boolean	**isEmpty**() Returns true if this collection contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this collection.
boolean	**remove**(Object o) Removes a single instance of the specified element from this collection, if it is present (optional operation).
int	**size**() Returns the number of elements in this collection.

java.util
Interface Iterator<E>

`public interface` **`Iterator<E>`**

An iterator over a collection.
This interface is a member of the Java Collections Framework.

Since:

1.2

See Also:

`Collection`, `ListIterator`, `Enumeration`

Method Summary	
`boolean`	**`hasNext`**`()` Returns `true` if the iteration has more elements.
`E`	**`next`**`()` Returns the next element in the iteration.
`void`	**`remove`**`()` Removes from the underlying collection the last element returned by the iterator (optional operation).

java.util
Interface Iterable<T>

All Known Subinterfaces:

… Collection<E>, … List<E>, … Set<E>, SortedSet<E>

All Known Implementing Classes:

AbstractCollection, AbstractList, … AbstractSet, … ArrayList, … LinkedList, … TreeSet, Vector

---

`public interface` **`Iterable<T>`**

Implementing this interface allows an object to be the target of the "foreach" statement.

Since:

1.5

Method Summary	
`Iterator<T>`	**`iterator`**`()` Returns an iterator over a set of elements of type T.

java.util
Interface List<E>
All Superinterfaces:
Collection<E>, Iterable<E>
All Known Implementing Classes:
…, ArrayList, …, LinkedList, …, Vector

```
public interface List<E>
extends Collection<E>
```

Method Summary

boolean	**add**(E e) Appends the specified element to the end of this list (optional operation).
void	**add**(int index, E element) Inserts the specified element at the specified position in this list (optional operation).
void	**clear**() Removes all of the elements from this list (optional operation).
boolean	**contains**(Object o) Returns true if this list contains the specified element.
boolean	**equals**(Object o) Compares the specified object with this list for equality.
E	**get**(int index) Returns the element at the specified position in this list.
int	**hashCode**() Returns the hash code value for this list.
boolean	**isEmpty**() Returns true if this list contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this list in proper sequence.
E	**remove**(int index) Removes the element at the specified position in this list (optional operation).
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present (optional operation).
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element (optional operation).
int	**size**() Returns the number of elements in this list.