TOB - TDs

22 Janvier, 2024

Louis Thevenet

Table des matières

1. TD1

Exercise 1.1:

1. **Point cartésien** `(Abscisse, Ordonnée)`
   Point polaire `(Rayon, Angle)`
2. tout
3. format textuel

Exercise 1.2:

1.

Point
Requête
x : double y : double mod : double
arg : double distance (Point p) : double
Commande
translater set_x(x : double) set_y(y : double)
set_mod(mod : double) set_arg(arg : double) afficher

Exercise 1.3:

1.

Point
Requête
x : double y : double mod : double
arg : double distance (Point p) : double
Commande
translater set_x(x : double) set_y(y : double)
set_mod(mod : double) set_arg(arg : double) afficher
Point(x : double, y : double)
Point(mod : double, arg : double)

Exercise 1.4:

```
class Point {
    double mod;
    double arg;

    double mod;
    double arg;

    // ...

    set_x(double x) {
        this.x = x;
        this.mod = Math.sqrt(this.x * this.x + this.y * this.y);
        this.arg = Math.atan2(this.y, this.x);
    }
}
```

2. TD2

Exercise 2.1:

1.

Ensemble
Requête
nombres : List<int> bool estVide() bool appartient(n:int) int cardinal() int min()
Commande
ajouter(n : int) supprimer(n:int)

```
interface Ensemble {
    boolean estVide();
    boolean appartient(int n);
    int cardinal();
    int min();
    void ajouter(int n);
    void supprimer(int n);
}
```

2.

```
Ensemble ensemble = new Ensemble(MAX);
Tantque !ensemble.estVide Faire
    afficher(ensemble.min());
    pour k de 1 à MAX Faire
        ensemble.supprimer(ensemble.min() * k);
    fin pour
Fin Tantque
```

3. 1.

EnsembleTab
Attributs
nombres : List<int>
Méthodes
Constructeurs
EnsembleTab(int max)

EnsembleChaine
Attributs
suivant : EnsembleChaine valeur : int
Méthodes
Constructeurs
EnsembleChaine(int max)

impl.

impl.

Ensemble
Requête
nombres : List<int> bool estVide() bool appartient(n:int) int cardinal() int min()
Commande
ajouter(n : int) supprimer(n:int)

2. On utilise `implements`
3. • Cas Tableau

   On ajoute au tableau

   • Cas Liste Chainée

   On ajoute un maillon à la fin de la liste chainée
4. ```
   // Cas Tableau
   int min() {
       return this.nombres.get(0);
   }

   // Cas Liste Chainée
   int min() {
       return this.valeur;
   }
   ```
5. Le cas `List` est plus efficace car le cas Liste Chainée est plus coûteux pour ajouter un élément.

Exercise 2.2:

```
public interface Ensemble<TypeDonnee> {
    boolean estVide();
    boolean appartient(TypeDonnee n);
    int cardinal();
    TypeDonnee min();
    void ajouter(TypeDonnee n);
    void supprimer(TypeDonnee n);
}
```

3. TD3

Exercise 3.1:

1.

CompteSimple
Requête
solde : double
Commande
créditer(s : double) {Pre: s≥0 / post: solde = old_solde+s} débiter(s : double) {Pre: s≥0 / post: solde = old_solde−s}
CompteSimple(soldeInitial : double)
CompteSimple()

titulaire

Personne
Requête
nom : String prenom : String
estFemme() : boolean estHomme() : boolean
Commande
Personne(nom : String, prenom : String)

CompteCourant
Requête
solde : double
Commande
créditer(s : double) {Pre: s≥0 / post: solde = old_solde+s} débiter(s : double) {Pre: s≥0 / post: solde = old_solde−s}
afficherReleve() afficherReleveDebits()
afficherReleveCredits()
CompteCourant(soldeInitial : double)
CompteCourant()

historique

Historique
Requête
dots
Commande
dots
Historique()

2.

```
c = new CS(100)
testCrediter1():
    c.crediter(0)
    c.getSole() == 100

testCrediter2():
    c.crediter(100)
    c.getSole() == 200

testTitulaire():
    p = new Personne("Doe", "John")
    c = new CS(100, p)
    c.getTitulaire() eq p
```

4. TD4

Exercise 4.1:

1. • `java ClassePrincipale un ~> [(<um>)F`
   • `java ClassePrincipale "" ~>`

   ```
   ("")
   [(<IF]
   ```

   • `java ClassePrincipale x ~>`

   ```
   ("x")
   [(<xF ERREUR
   ```

2.

```
public static void main(String[] args) {
    double somme = 0;
    int counter = 0;
    for (String arg : args) {
        try {
            somme += Double.parseDouble(arg);
        } catch (NumberFormatException e) {
            counter++;
            System.out.println("ERREUR");
        }
    }
    System.out.println(somme);
    System.out.println("Nombre d'erreurs ignorées : " + counter);
}
```

3.

```
class LivretA extends CompteCourant {
    double taux;
    double plafond;

    LivretA(double soldeInitial, double taux, double plafond) {
        super(soldeInitial);
        this.taux = taux;
        this.plafond = plafond;
    }

    void capitaliser() {
        this.crediter(this.solde * this.taux);
    }
}
```
```