

TOB - TDs

22 Janvier, 2024

Louis Thevenet

Table des matières

1. TD1	2
2. TD2	2
3. TD3	2
4. TD4	2
5. TD9	2

1. TD1

Exercice 1.1:

1. Point cartésien (Abscisse, Ordonnée)
Point polaire (Rayon, Angle)

2. tout

3. format textuel

Exercice 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

Exercice 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)		

Exercice 1.4:

1.

```
1 class Point {
2     double mod;
3     double arg;
4
5     double mod;
6     double arg;
7
8     // ...
9
10    set_x(double x) {
11        this.x = x;
12        this.mod = Math.sqrt(this.x * this.x + this.y * this.y);
13        this.arg = Math.atan2(this.y, this.x);
14    }
15 }
```

2. TD2

Exercice 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)	

```
1 interface Ensemble {
2     boolean estVide();
3     boolean appartient(int n);
4     int cardinal();
5     int min();
6     void ajouter(int n);
7     void supprimer(int n);
8 }
```

2.

```
1 Ensemble ensemble = new Ensemble(MAX);
2 Tantque !ensemble.estVide Faire
3     afficher(ensemble.min());
4     pour k de 1 à MAX Faire
5         ensemble.supprimer(ensemble.min() * k);
6     fin pour
7 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.

Exercice 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

Exercice 3.1:

1.

CompteSimple	
Requête	
solde : double	
Commande	
créditer(s : double) {Pré: s ≥ 0 post: solde = old_solde + s	débiter(s : double) {Pré: 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)	

historique

Historique	
Requête	
dots	
Commande	
dots	
Historique()	

2.

```
1 c = new CS(100)
2 testCrediter1():
3     c.crediter(0)
4     c.getSole() == 100
5
6 testCrediter2():
7     c.crediter(100)
8     c.getSole() == 200
9
10 testTitulaire():
11     p = new Personne("Doe", "John")
12     c = new CS(100, p)
13     c.getTitulaire() eq p
```

4. TD4

Exercice 4.1:

1. • java ClassePrincipale un ~> [(<un>)F
• java ClassePrincipale "" ~>
 ("")
 [(<F)]
• java ClassePrincipale x ~>
 ("x")
 [(<x>F ERREUR

2.

```
1 public static void main(String[] args) {
2     double somme = 0;
3     int counter = 0;
4     for (String arg : args) {
5         try {
6             somme += Double.parseDouble(arg);
7         } catch (NumberFormatException e) {
8             counter++;
9             System.out.println("ERREUR");
10        }
11    }
12    System.out.println(somme);
13    System.out.println("Nombre d'erreurs ignorées : " + counter);
14 }
```

3.

```
1 class LivretA extends CompteCourant {
2     double taux;
3     double plafond;
4
5     LivretA(double soldeInitial, double taux, double plafond) {
6         super(soldeInitial);
7         this.taux = taux;
8         this.plafond = plafond;
9     }
10
11     void capitaliser() {
12         this.crediter(this.solde * this.taux);
13     }
14 }
```

5. TD9

Exercice 5.1:

1.

Monnaie	
Requête	
valeur : Integer	devise : String
Commande	
Ajouter(m: Monnaie)	
Retirer(m: Monnaie)	
Monnaie(valeur : int, devise : String)	

throws

`DeviseInvalideException`	
Exception	
valeur : Integer	devise : String

```
1 public class Monnaie() {
2     int valeur;
3     String devise;
4
5     public Monnaie(int vlauer, String devise) {
6         this.valeur=valeur;
7         this.devise = devise;
8     }
9
10    private Boolean MemeDevise(Monnaie m) {
11        return devise== m.devise;
12    }
13
14    public Ajouter(Monnaie m) throws DeviseInvalideException {
15        if (m.devise!= devise)
16            throw new DeviseInvalideException();
17        valeur += m.valeur;
18    }
19
20    public Retirer(Monnaie m) throws DeviseInvalideException {
21        if (m.devise != devise)
22            throw new DeviseInvalideException();
23        valeur -= m.valeur;
24    }
25
26 }
27 }
```

2