FICHE FINALE MAPD – UML / JAVA / OOP

---

1) UML – Relations essentielles

Héritage : A extends B = A est un B
Interface : A implements I = A respecte un contrat
Association : lien simple (utilise)
Composition : losange noir = fait partie de
Agrégation : losange blanc = contient sans dépendance

---

2) Pattern Composite (clé du sujet)

```
interface ProcessStep { void execute(Part p); }
class Operation implements ProcessStep {
public void execute(Part p) { build(p); }
public void build(Part p) { ... }
}
class Process implements ProcessStep {
private List steps = new ArrayList<>();
public void add(ProcessStep s) { steps.add(s); }
public void execute(Part p) {
for (ProcessStep s : steps)
s.execute(p);
}
}
```

---

3) Exemples importants

```
class Painting extends Operation {
private String color;
public Painting(String color) { this.color = color; }
}
class PaintingNLayers extends Process {
private String color;
private int layers;
public PaintingNLayers(String color, int layers) {
this.color = color;
this.layers = layers;
for (int i = 0; i < layers; i++)
this.add(new Painting(color));
}
}
```

---

4) Stack (pile) – Génériques

```
public interface Stack {
void push(T v);
T pop();
boolean isEmpty();
}
class ArrayStack implements Stack {
private ArrayList data = new ArrayList<>();
public void push(T v) { data.add(v); }
```

```
public T pop() { return data.remove(data.size() - 1); }
public boolean isEmpty() { return data.isEmpty(); }
}
```

---

5) Notions Java essentielles

Encapsulation : attributs privés, getters/setters
final : immuable / non héritable / non overridable
== compare références ; equals() compare valeurs
Référence = adresse de l'objet
Exceptions fréquentes :
- IllegalArgumentException : argument invalide
- NullPointerException : objet null
- IndexOutOfBoundsException : index hors limites
- IllegalStateException : état incohérent
- UnsupportedOperationException : action non permise

---

6) Exercice 1 – Notions utiles

---

requires : précondition (ce qui doit être vrai AVANT)
ensures : postcondition (ce qui doit être vrai APRÈS)
throw : signale une erreur de logique d’utilisation
Enum : type restreint de constantes (ex: ItemKind.BOOK)
Composition : un objet appartient entièrement à l'autre
Encapsulation : protéger les données internes
Setter : modifie un attribut (souvent void)

---

7) Main – modèle complet

```
public class Main {
public static void main(String[] args) {
Part part = new Part("raw");
Process process = new Process();
process.add(new Painting("red"));
process.add(new PaintingNLayers("blue", 3));
process.execute(part);
}
}
```

Mini résumé express :
Composite = interface + Process + Operation
Generics =
ArrayList = O(1) push/pop
== ref, equals valeur
extends = héritage, implements = contrat
