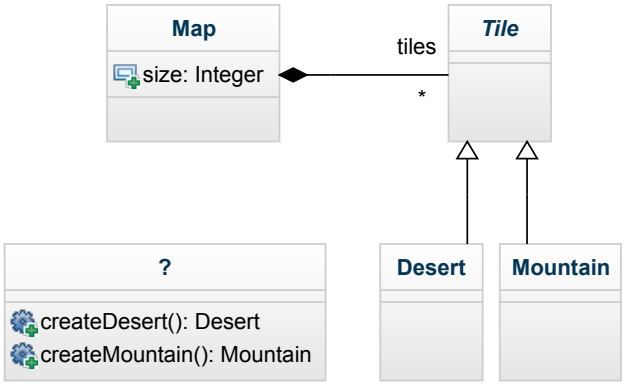
Patron de conception / Design pattern – 5 EII

TP3

1 Exam exercice 1

A *Map* is composed (composition *tiles*) of *Tiles* : *Mountain* and *Desert*. A *Map* is square and its size is defined by the attribute *size*. The position (x, y) of a tile in a map can be computed using the position of the tile in the liste *tiles* and the size of the map.



Question 1. What is the name of the design pattern behind the unnamed class (the class named " ?") ? Give a name to this class to be reused in the following questions.

Imagine you have to create a very large map. You thus want to optimise the memory by limiting the number of created *Tile* instances.

Question 2. What is the name of the design pattern to use in this case ?

Question 3. Explain why this design pattern is useful here.

Question 4. Apply this design pattern to the class diagram provided above.

Question 5. Once this design pattern used, given the Java code (or the pseudo-code) of the unnamed class (the one you gave a name in question 1).

Imagine you have three algorithms for creating maps (e.g. a map mainly composed of *Montain* tiles).

Question 6. What is the name of the design pattern to use in this case ?

2 Exam exercice 2

2.1 Question

Quel est le patron de conception utilisé dans le code suivant ? Justifiez.

```
public interface Object {
  void method();
}

public class ObjectImpl1 implements Object{
  @override public void method() {
    ...
  }
}

public class ObjectImpl2 implements Object{
  Object obj = new ObjectImpl1();

  @override public void method() {
    // Checking that some conditions are respected before calling 'method'.
    if(...)
      obj.method();
  }
}
```

2.2 Patron de conception poids-mouche

L'ADN se compose de 4 bases azotées différents : A, G, C et T. On vaudrait concevoir un programme orienté objet capable de charger en mémoire de l'ADN ce qui implique de pouvoir gérer un grand nombre de bases azotées.

1. Modéliser (diagramme de classes UML) l'ADN, les bases azotées ainsi que tout moyen permettant une gestion minimale des instances des bases azotées grâce au patron de conception poids-mouche.

2. Donner le code Java de la fabrique associée au poids-mouche.