Tutorat Info 2021 - JAVA

2 - Exemple de modélisation Objet

Markus Person a encore besoin de vous ! Saurez-vous l'aider à créer Minecraft en Java ? (Comment ça c'est la même chose que la dernière fois ?)

Exercice 1 - Entités

Minecraft est composé d'entités. Une entité c'est n'importe quelle chose qu'on peut trouver dans le monde de Minecraft : un bloc, un monstre, un joueur...

Plus précisément, c'est n'importe quelle chose qu'on peut trouver dans l'espace en trois dimensions du jeu.

Une entité a quelques propriétés simples :

- Un identifiant unique (par exemple 1 ou 0 ou 135) pour pouvoir la retrouver plus tard. Cet identifiant commence à 0 et augmente pour chaque entité crée.
- Une position dans l'espace en trois dimensions (par exemple (1, 3, 5) ou (16.5, 18.38, -1357)) pour savoir où elle est.

Consignes : Codez la classe `Entity` (ça veut dire entité en anglais), ses `setters` et ses `getters` (sauf pour l'identifiant qui ne doit jamais changer)

Exercice 2 - Plusieurs types d'entités

Minecraft est composé aussi bien de créatures (personnage du joueur, monstres) que de blocs. Et ils n'ont a peu près rien à voir à part qu'ils sont tous dans l'espace en trois dimensions.

Par exemple, les blocs peuvent être cassés ou posés. Mais les créatures peuvent bouger, attaquer et mourir.

Quand on a ce genre de cas de figure en Java on peut se servir de l'héritage. Il va nous permettre de partager des choses entre plusieurs objets. Ici on va partager le fait d'être une entité entre les créatures et les blocs.

Dans Java grâce à l'héritage :

- On pourra dire qu'un bloc est une entité et donc faire tout ce que peut faire l'entité avec.
- On pourra dire qu'une créature est une entité et donc faire tout ce que peut faire l'entité avec.

- Par contre on ne pourra pas forcément dire l'inverse, car en tombant sur une entité au hasard on n'aura aucune garantie qu'il s'agit plutôt d'un bloc que d'une créature et vice-versa.

Consignes : 1. créez les classes `Creature` et `Block` héritant d'`Entity` 2. créez une classe avec un `main` 3. testez de déclarer une variable de type `Entity` mais de lui assigner un bloc ou une créature à la place 4. essayez de transformer une entité (qui en réalité est un bloc ou une créature) en un bloc ou une créature

Correction exercice 1

Version 1 : propriétés et initialisation avec un constructeur

```java
//Entity.java
public class Entity {
    private static int nbEntities = 0;

    private final int id;
    private float x, y, z;
    // x = position axe NORD/SUD (dimension n°1)
    // y = position axe EST/OUEST (dimension n°2)
    // z = position axe HAUT/BAS (dimension n°3)

    public Entity(float x, float y, float z) {
        id = nbEntities;
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public int getId() {
        return id;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public int getZ() {
        return z;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }
```

```
    public void setZ(int z) {
        this.z = z;
    }
}
```

Version 2 (optionel) : stockage des entités selon leur `id` dans la partie statique

```
//Entity.java
public class Entity {
    private static int nbEntities = 0;
    private static Map<Integer, Entity> entities = new HashMap<Integer, Entity>();

    private final int id;
    private float x, y, z;

    public Entity(float x, float y, float z) {
        id = nbEntities;
        this.x = x;
        this.y = y;
        this.z = z;

        entities.put(id, this);
    }

    //... getters et setters définis dans la version 1

    //Obtenir une entité par son identifiant
    public static Entity getById(int id) {
        return entities.get(id);
    }

    //Obtenir toutes les entités dans une liste
    // nb: map.values() renvoie une collection d'entités (pas tout à fait comme
une liste mais on est pas loin), c'est pourquoi on recrée une liste par dessus
    public static List<Entity> getAll() {
        return new ArrayList<Entity>(map.values());
    }
}
```

Correction exercice 2

```
//Block.java
public class Block extends Entity {

}
```

```

```

```java
//Creature.java
public class Creature extends Entity {

}
```

```java
//Main.java
public class Main {
    public static void main(String[] args) {
        Entity e1 = new Entity(1.35, 2.1, 4.84);
        System.out.println(e1.getId());
        System.out.println(String.format(
            "{%s, %s, %s}",
            e1.getX(),
            e1.getY(),
            e1.getZ()
        ));

        Entity e2 = new Creature(58.9, 4.8, 5.18);
        System.out.println(e2.getId());
        System.out.println(String.format(
            "{%s, %s, %s}",
            e2.getX(),
            e2.getY(),
            e2.getZ()
        ));

        Entity e3 = new Block(8.17, 454.97, 4.11);
        System.out.println(e2.getId());
        System.out.println(String.format(
            "{%s, %s, %s}",
            e3.getX(),
            e3.getY(),
            e3.getZ()
        ));

        for(Entity e: Entity.getAll()) {
            System.out.println(e);
        }

        System.out.println(Entity.getById(2));

        System.out.println(e2 instanceof Entity);
        System.out.println(e2 instanceof Creature);
        System.out.println(e2 instanceof Block);

        Creature e2_version_creature = (Creature) e2;

        try {
            Creature e3_version_creature = (Creature) e3;
        } catch(Exception e) {
            System.out.println("Houla, j'ai planté car e3 n'est pas une créature
```

```
!")
        }
    }
}
```