Abstract Factory

A factory of factories

Sometimes we want a factory for related stuff

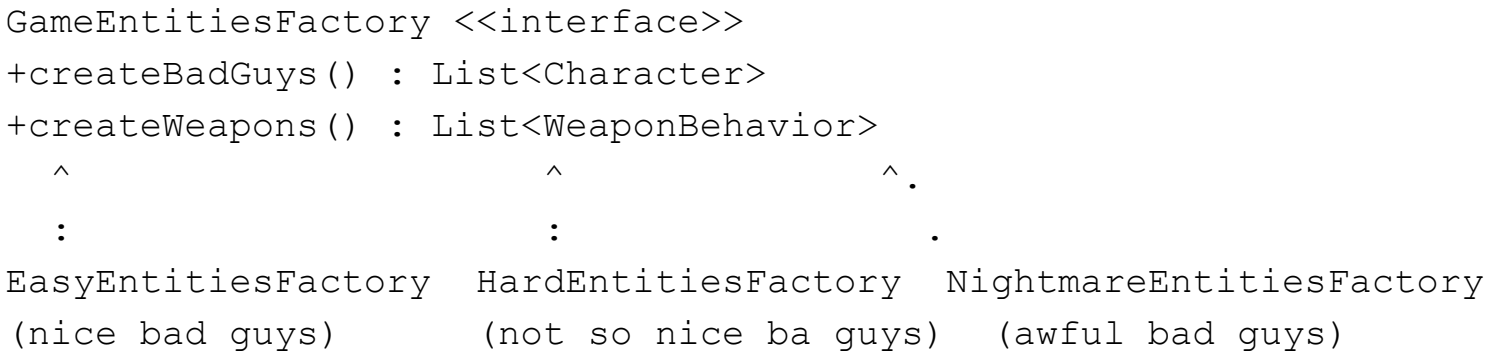
Perhaps it would be nice to have a factory for our adventure game which could help us both create `Character`s and `WeaponBehavior`s.

It would be nice to get all characters at once, and all weapons needed.

We could have a factory which could produce both weapons and characters, for different levels of the game.

Characters will be the bad guys the player encouters, and weapons will be the weapons you may find as you walk around the game.

A factory of related stuff



Let’s look at some code

```
import java.util.List;
public interface GameEntitiesFactory{
  public List<Character>createBadGuys();
  public List<WeaponBehavior>createWeapons();
}
```

Let’s look at some code

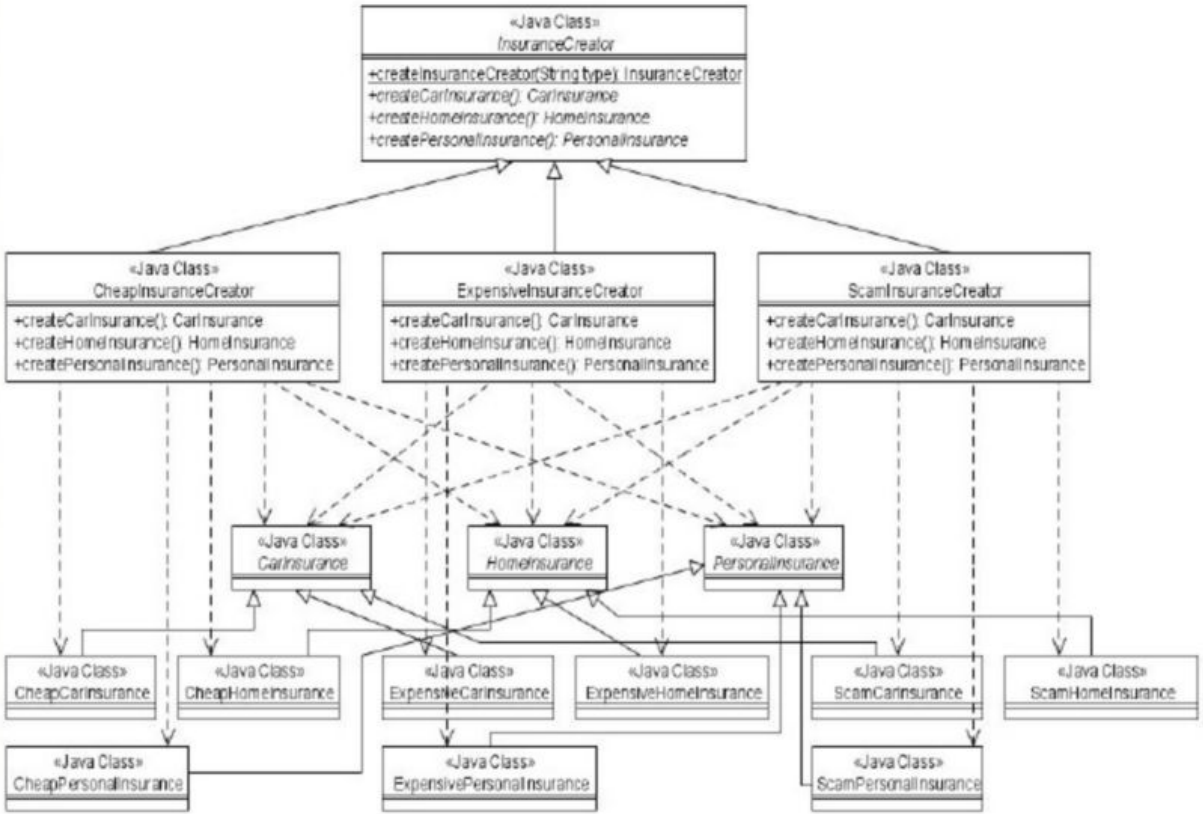
```
public class EasyEntitiesFactory implements GameEntitiesFactory{
//...

  // Three knights and one troll
  public List<Character>createBadGuys(){
    List<Character> badGuys = new ArrayList<>();
    badGuys.add(knightFac.createCharacter(WeaponType.SWORD, "Sir Humpty"));
    badGuys.add(knightFac.createCharacter(WeaponType.SWORD, "Sir Dumpty"));
    badGuys.add(knightFac.createCharacter(WeaponType.SWORD, "Sir James"));
    badGuys.add(trollFac.createCharacter(WeaponType.CLUB, "Trolly"));
    return badGuys;
  }
```

AbstractFactory explained by WM-data/Logica CC-SA 3.0

Structure

Below you see the class diagram of the following small example.



Any questions?

https://www.scribd.com/document/9973578/Design-Patterns-Explained-With-Java-and-Uml2-2008

Let’s look at some code

```
// HardEntitiesFactory
  public List<Character>createBadGuys(){
    List<Character> badGuys = new ArrayList<>();
    badGuys.add(knightFac.createCharacter(WeaponType.SWORD, "Sir Humpty"));
    badGuys.add(knightFac.createCharacter(WeaponType.SWORD, "Sir Dumpty"));
    badGuys.add(orchFac.createCharacter(WeaponType.CLUB, "Sir James"));
    badGuys.add(trollFac.createCharacter(WeaponType.CLUB, "Trolly"));
    return badGuys;
  }
```

Test game

Idea: Loop through all the bad guys and have the player fight them.

After each fight, the player finds a new weapon if there is any

Also, the player finds a magic potion which gives some health

Setting up the game

```
CharacterFactory cFactory = new KnightFactory();
Character player =
  cFactory.createUnarmedCharacter("Sir Playsalot");
// Test level Easy:
GameEntitiesFactory gef = new EasyEntitiesFactory();
List<Character>    opponents = gef.createBadGuys();
List<WeaponBehavior> weapons = gef.createWeapons();
int weaponIndex=0;
```

Outer loop

```
for(Character c : opponents){
  System.out.println("<<<Oh no, not a bad guy!");
  System.out.println(player + " meets " + c + " who attacks!");
  // fight until death!
}
```

Fight loop

```
for(Character c : opponents){
  System.out.println("<<<Oh no, not a bad guy!");
  System.out.println(player + " meets " + c + " who attacks!");
  while(c.health()>0){ // Player never dies :-P
    c.fight(player);
    player.fight(c);
    player.fight(c);
    player.fight(c);
  }
  // Find a weapon!
}
```

Find next weapon and upgrade

```
for(Character c : opponents){
  System.out.println("<<<Oh no, not a bad guy!");
  System.out.println(player + " meets " + c + " who attacks!");
  while(c.health()>0){ // Player never dies :-P
    // fight loop
  }
  // Find a weapon!
  if(weaponIndex < weapons.size()){
    System.out.println(player + " finds " + weapons.get(weaponIndex)
                               + " and upgrades his weapon!");
    player.changeWeapon(weapons.get(weaponIndex));
    weaponIndex++;
  }
}
```

We could use a factory to get the factory

```
// Use a simple factory to get the game entities factory
GameEntitiesFactory gef = Levels.getEntetiesFactory(Levels.EASY);
// would return a new EasyEntitiesFactory();


// This way, we have decoupled the game from knowing the names of
// the concrete EntitiesFactories
```

End!

Please do the exercises and read the pages linked to under “Further reading”