Factories

More creational patterns
Variations on a theme

The factory pattern exists in a few different flavors. We’ll look at three of them.

- Simple Factory
- Factory Method
- Abstract Factory
Simple Factory - motivation

```java
public class Game{
    public static void main(String[] args){
        Troll troll = new Troll("Lillfjant", new Club());
        Knight sirJames = new Knight("Sir James", new Sword());
        Knight blackKnight = new Knight("Fistfighting Black Knight");
    }
}
```

- Sensitive to changes!
  - Class name changes?
  - We need to know how the constructors work (if that changes, we'll have to change too)
  - We need to know about Club and Sword
  - Overloaded constructors lack undescrptive names
Simple Factory - Design

```
+--------------------------------------------+        <<enum>> CharacterType
|                 <<interface>>              |        <<enum>> WeaponType
|               CharacterFactory             |        
+--------------------------------------------+        ^
|                                          :        :
| +createCharacter(CharacterType,           :        :
|      WeaponType,                          :        :
|      String) : Character                  :        
+--------------------------------------------+        ^
|                                          :        :
|                                          :        :
| +--------------------------------------------+        SimpleCharacterFactory
|                                          +--------------------------------------------+
|                                          | +createCharacter(CharacterType,           |
|                                          |      WeaponType,                          |
|                                          |      String) : Character                  |
+--------------------------------------------+
```
Simple Factory - Design

+--------------------------------------------+
|                 <<interface>>              |        <<enum>> WeaponType
|                 WeaponFactory              |
+--------------------------------------------+

| +createWeapon(WeaponType) : WeaponBehavior |
+--------------------------------------------+

| SimpleWeaponFactory                      |
+--------------------------------------------+

| +createWeapon(WeaponType):WeaponBehavior  |
+--------------------------------------------+
public class Game{
    public static void main(String[] args){
        CharacterFactory cFactory = new SimpleCharacterFactory();
        Character troll =
            cFactory.createCharacter(CharacterType.TROLL,
                               WeaponType.CLUB,
                               "Lillfjant");

        Character sirJames =
            cFactory.createCharacter(CharacterType.KNIGHT,
                               WeaponType.SWORD,
                               "Sir James");

        Character blackKnight =
            cFactory.createCharacter(CharacterType.KNIGHT,
                               WeaponType.UNARMED,
                               "Fistfighting Black Knight");
    }
    }
    // Creation is decoupled from the client
Simple Factory - Benefits

Where do we have code to change if we

- Create a new concrete character Class
- Create a new WeaponBehavior
- Change the constructor of Character?
- Change the constructor of a Weapon?
Simple Factory - Benefits

Where do we have code to change if we

- Create a new concrete character Class
  - CharacterType (enum)
  - SimpleCharacterFactory - createCharacter
- Create a new WeaponBehavior
  - WeaponType (enum)
  - SimpleWeaponFactory - createWeapon
- Change the constructor of Character?
  - SimpleCharacterFactory - createCharacter
- Change the constructor of a Weapon?
  - SimpleWeaponFactory - createWeapon

Client code will not break.
Simple Factory - Benefits

We can add a method to the factory which will be more descriptive than an overloaded constructor.
Method for creating an unarmed character:

```java
public Character createUnarmedCharacter(CharacterType character, String name){
    return createCharacter(character, WeaponType.UNARMED, name);
}
```
SimpleFactory - more client code

```java
CharacterFactory cFactory = new SimpleCharacterFactory();
Character troll =
    cFactory.createCharacter(CharacterType.TROLL,
                             WeaponType.CLUB,
                             "Lilljfjant");
Character sirJames =
    cFactory.createCharacter(CharacterType.KNIGHT,
                             WeaponType.SWORD,
                             "Sir James");
Character blackKnight =
    cFactory.createCharacter(CharacterType.KNIGHT,
                             WeaponType.UNARMED,
                             "Fistfighting Black Knight");
Character evilOrch =
    cFactory.createCharacter(CharacterType.ORCH,     // new character type!
                             WeaponType.SHOTGUN,
                             "Ugly scary Orch");
Character unarmedOrch =
    cFactory.createUnarmedCharacter(CharacterType.ORCH,     // new factory method!
                                     "Ugly scary unarmed Orch");
```
Simple Factory - Problems

- Still a lot of work to add new types
  - we have to change both the enum and the factory method
- Can you see any other problems?