What are design patterns?

No, really, what are they?

Contents:

- Design patterns
- Different categories
- Design patterns have...
- Object oriented design principles
- Knowing the principles is not enough
- Design patterns are...
- Example design pattern (Strategy)
    - Principle - encapsulate what varies (and make it interchangeable in runtime)

Design patterns

- Are discovered, not invented
- Describe solutions to common problems in very general terms
- Originally came from architecture (physical architecture)
- Form a catalogue of solutions (with examples)

Different categories of patterns

- Creational
- Structural
- Behavioral

Design patterns have/describe

- A name
- A context
- A problem (what it tries to solve)
- A solution - how to solve the problem

In object oriented design

When designing systems and components in an object oriented language we try to achieve some goals according to some principles.

We'll talk more about principles in another lecture but examples of principles include:

- Don't repeat yourself
- Avoid coupling between components (they should be independent)
- Encapsulate what varies (e.g. how is something done?)
- Favor composition over inheritance
- Program to interfaces (supertypes), not implementations
- … and many more

Knowing the principles by name is not enough

You might know the basic principles of OO by name and basic ideas, but it doesn’t always help.

You really want to build flexible designs, which are easy to maintain because they were built knowing that things will change in the future, so they cope well with these changes.

Design patterns are a catalog of ways to achieve the above, expressed in general terms.

Design patterns are...

- Recurring solutions to design problems
- Practical solutions to real problems (not just theories)
- Discovered, because they are used for real by many people
- Trying to balance trade-offs for a particular challenge
- Documented “best practices”
- Based on experience and documented for others to learn
- Really very overhyped (lots of fan clubs)
- Often quite misunderstood (we’ll try our best to help you)
- Not always the perfect solution to *your problem*
    - But often a good start

Example problem - the adventure game

We’re writing a text-based adventure game and identify the following classes:

Character - abstract base class for all characters in the game

Knight - one subtype of Character

Troll - another subtype of Character

Characters can fight each other, inflicting damage (reducing health)

Characters can have different weapons with different damage capacity

What design principle could we apply?

What differs between characters (other than name)?

- The weapon (and therefore the damage they can inflict on others)

If we apply the principle “Encapsulate what differs”, how do we encapsulate the way characters attack/use their weapon?

What should fight() do?

We decide that all Characters (regardless of subtype) should have the ability to fight another Character:

public abstract void fight(Character opponent);

What should the result be?

The opponent should lose health, depending on the damage of the weapon.

How to represent the instance variable weapon?

What type should we use to represent the weapon (indirectly affecting the fight behavior)?

Suggestions?

What should the fighting behavior be if the character has no weapon?

What if we want characters to change weapon?

It would be nice if a Character in the game can lose one weapon and pick up another weapon during runtime (when the game is running).

How can we implement this in the Character class?

Applying the strategy pattern

Strategy pattern:

The strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

Head First Design Patterns

What algorithm (behavior) should we encapsulate?

It’s the behavior of the use of weapon which varies between characters (at least that’s the behavior we focus on now).

So, some kind of WeaponBehavior, then.

It is also this behavior we’d like to change regardless of what character type we are talking about.

So, each Character should have a weapon (or none), and they should have the ability to change weapon.

Weapon is what we’d like to encapsulate (into each own type).

What is a weapon?

Anything that can be used to attack someone with could be regarded as a weapon.

Weapons are then defined by their ability (behavior) to be used as a weapon.

A book can be used as a weapon, as well as a magazine or a knife.

en:User:Majorly

Millwall brick held

CC BY 2.5

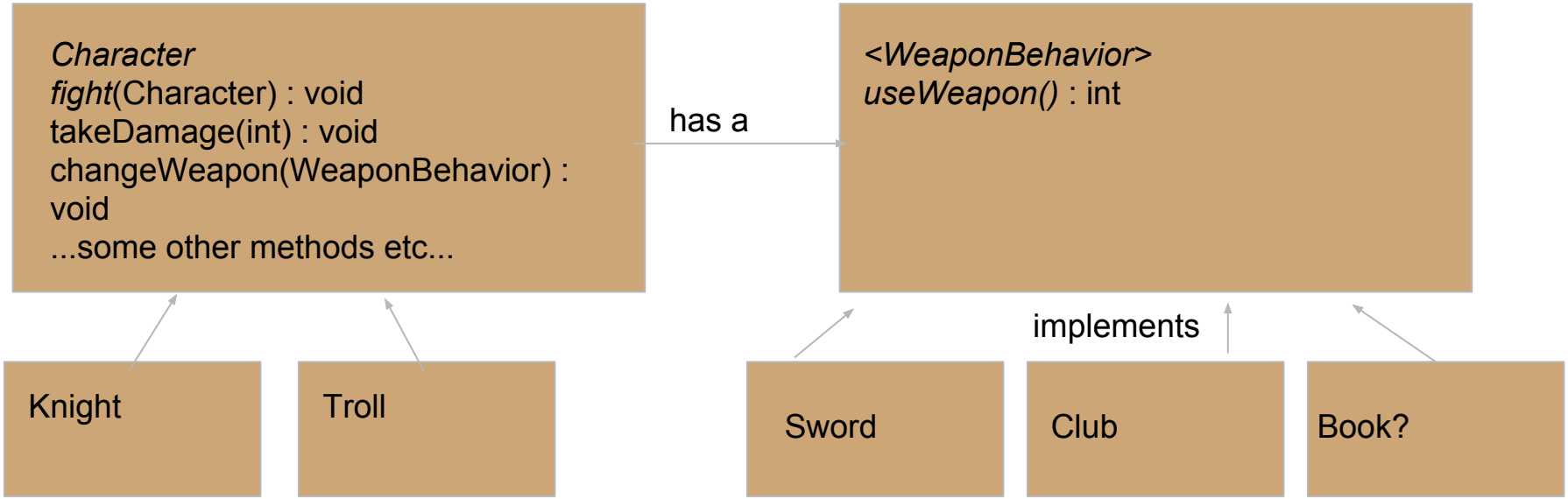
When the defining property is an ability...

Use an interface!

Suggestion:

```
public interface WeaponBehavior{
  // Returns the damage health points of this weapon
  public int useWeapon();
}
```

Overview



Good thing is that the classes for weapons can simply implement the interface but belong to other class hierarchies.

Simulation of the adventure game

<event>Sir James with health: 100 carrying Excalibur runs into the troll Lillfjant with health: 100 carrying Ugly wooden club
<event>The troll attacks
-Bränn, bränn, bränn!
Lillfjant uses Ugly wooden club against Sir James
The club makes a swooshing sound
Score after attack: Lillfjant health: 100 carrying Ugly wooden club - Sir James health: 80 carrying Excalibur

<event>The knight returns the favor
-None shall pass!
Sir James nobly uses its Excalibur against the simple Lillfjant
The sword shines and cuts through the air before it hits its target
Score after attack: Sir James health: 80 carrying Excalibur - Lillfjant health: 85 carrying Ugly wooden club

<event>The knight realizes that the sword isn't enough

Simulation of the adventure game, cont.

<event>The knight finds a shotgun, drops the sword and picks the shotgun up
<event>The troll attacks again
-Bränn, bränn, bränn!
Lillfjant uses Ugly wooden club against Sir James
The club makes a swooshing sound
Score after attack: Lillfjant health: 85 carrying Ugly wooden club - Sir James with health: 60 carrying Shotgun

<event>The knight fires the shotgun
-None shall pass!
Sir James nobly uses its Shotgun against the simple Lillfjant
Schklikt, klikt Ka-POW goes the shotgun
Score after attack: Sir James with health: 60 carrying Shotgun - Lillfjant with health: 40 carrying Ugly wooden club

<event>The knight shoots again
-None shall pass!
Sir James nobly uses its Shotgun against the simple Lillfjant
Schklikt, klikt Ka-POW goes the shotgun
Score after attack: Sir James health: 60 carrying Shotgun - Lillfjant with health: -5 carrying Ugly wooden club

Simulation of the adventure game, cont.

<event>The evil, but unarmed Black Knight shows up and attacks Sir James
Hitting and kicking (in lack of weapons)!
Score after attack: Fistfighting Black Knight health: 100 - Sir James health: 55 carrying Shotgun

<event>Sir James gets tired of this bad day and fires three rounds against Black Knight
-None shall pass!
Sir James nobly uses its Shotgun against the simple Fistfighting Black Knight
Schklikt, klikt Ka-POW goes the shotgun
Score after attack: Sir James health: 55 carrying Shotgun - Fistfighting Black Knight health: 55
-None shall pass!
Sir James nobly uses its Shotgun against the simple Fistfighting Black Knight
Schklikt, klikt Ka-POW goes the shotgun
Score after attack: Sir James health: 55 carrying Shotgun - Fistfighting Black Knight health: 10
-None shall pass!
Sir James nobly uses its Shotgun against the simple Fistfighting Black Knight
Schklikt, klikt Ka-POW goes the shotgun
Score after attack: Sir James health: 55 carrying Shotgun - Fistfighting Black Knight health: -35