

Object Pool

- Il design pattern Factory Method può implementare un object pool
- Object Pool
 - Qualche volta è utile poter riusare le istanze già create, anziché crearne nuove, ovvero una nuova per ciascuna richiesta
 - Poiché in alcuni casi, istanziare ed inizializzare sono operazioni lente
 - Un object pool è un repository che gestisce le istanze già create, una istanza sarà estratta dal pool quando un client ne fa richiesta
 - Il pool può crescere o può avere dimensioni fisse
 - Dimensioni fisse: se non ci sono oggetti disponibili al momento della richiesta, non ne creo di nuovi
 - Il client restituisce l'istanza usata al pool quando non più utile
 - Il Factory Method è ottimo per gestire un pool di oggetti
 - I client fanno richieste, come visto prima per il Factory Method
 - I client dovranno dire quando l'istanza non è più in uso, quindi riutilizzabile
 - Lo stato dell'istanza da riusare potrebbe dover essere ri-scritto
 - L'object pool dovrebbe essere unico -> uso un Singleton

E. Tramontana Pool, Dependency - 12 May 10 1

Dependency Injection

- Il design pattern Factory Method può essere usato per inserire le dipendenze necessarie ad altri oggetti
- Dependency injection
 - Una classe che usa un servizio *dipende* dal servizio
 - Come faccio conoscere alla classe il servizio da cui dipende?
- Esempio di dependency
 - Una classe `TextEditor` usa un servizio `SpellingCheck`
 - `TextEditor` dipende dal servizio `SpellingCheck`
 - Ci sono tante classi che implementano il servizio `SpellingCheck`, in base alla lingua usata: `SpellingCheckEnglish`, `SpellingCheckItalian`, etc.
 - `TextEditor` deve poter essere collegato ad una delle classi `SpellingCheck`

E. Tramontana Pool, Dependency - 12 May 10 3

Esempio codice Object Pool

```
import java.util.LinkedList;
// CreatorPool is a ConcreteCreator and implements an object pool
public class CreatorPool extends ShapeCreator {
    private LinkedList<Shape> pool = new LinkedList<Shape>();
    public Shape getShape() {
        Shape s;
        if (pool.size() > 0) s = pool.remove();
        else s = new Circle();
        return s;
    }
    public void releaseShape(Shape s) {
        pool.add(s);
    }
}
```

E. Tramontana Pool, Dependency - 12 May 10 2

Esempio Codice Dependency Injection

```
public class TextEditor {
    private SpellingCheck speller;
    public TextEditor(SpellingCheck sp) {
        speller = sp;
    }
    public void put(String s) {
        // receive input
        if (speller.check(s)) ...
        else ...
    }
}

public class CreatorText {
    public static TextEditor getEnglishEditor() {
        return new TextEditor(new SpellingCheckEnglish());
    }
    public static TextEditor getItalianEditor() {
        return new TextEditor(new SpellingCheckItalian());
    }
}
```

E. Tramontana Pool, Dependency - 12 May 10 4

Note sul codice

- Considerazioni
 - L'attributo `speller` di `TextEditor` è inizializzato, attraverso la chiamata al costruttore, con l'opportuna istanza
 - Il metodo `Factory` conosce la classe per lo spelling da usare ed inserisce (inject) la dipendenza sulla classe `TextEditor`
 - All'interno della classe `TextEditor` non è fissato l'uso di una classe `SpellingCheck`
 - Posso sostituire la classe `SpellingCheck` facilmente