Indice

1 Design pattern

1.1 Template pattern (behaviour)

Crei una classe astratta con metodi final la quale rappresenta un modo generale di fare una certa cosa (Game) poi utilizzando una class generale di dimostrazione (Template Demo), istanzi delle classi concrete che estendono la classe astratta generale (Game–¿ Calcio, Game –¿ Cricket) e dimostri che funziona

1.2 Singleton pattern (creational)

1.3 Proxy (structural)

Una classe fa da proxy per accedere ad un altra classe (ImageProxy fornisce Image)

1.4 Observer(behaviour)

Usato quand eiste una relazione 1:m dove m oggetti devono eseguire compiti se l'1 ha un cambiamento di qualche tipo. Tre classi attore: Subject-Observer-Client. Subject ha un metodo per aggiungere e togliere osservatori Subject ha un metodo setState(), dentro di esso viene richiamtato il metodo notifyAllObservers()

1.5 Iterator()

Popolare in Java e .net,usato per accedere in maniera sequenziale agli elementi di di una collezzione senza conoscere come tale meccanismo viene realizzato. Classe Container e Iteratore, la prima ha come membro una lista di oggetti da iterare e il metodo getIterator, la seconda ha un indice che corrisponde l'elemento corrente della lista

1.6 Factory (Creational)

Classe principale concreta chiamata ObjectFactory che contiene il metodo getObject(informations) dove Object è il tipo di oggetto che quella Factory genera e fornisce al chiamante in base alle informazioni date alla getObject

1.7 Facade (structural)

Una classe nasconde le complessità di altre classi e fornisce un'interfaccia semplice al client

1.8 Decorator (strctural)

La classe Decorator wraps un'altra classe alla quale aggiunge funzionalità senza alterla Es: ShapeDecorator viene implementato da RedShepDecorato il quale qualsiasi shape abbia, la setta rossa prima di draw()

1.9 Data Access Object (DAO) Pattern

Separa l'accesso ai dati tramite operazioni low level da funzioni di alto livello Multiple classi studente, StudentDao fornisce funzioni come getStudent(Student s), getAllStudents() etc

1.10 Abstract Factory

Soggetto del pattern è una super factory che genera altre factory dato un certo oggetto

2 Diagrammi

2.1 Caso d'uso

- Il diagramma UML dei casi d'uso è un tool per la modellazione del comportamento di un sistema.
- Descrive gli attori che interagiscono con il sistema, cosa fanno, e cosa ottengono dal sistema.
- A questo punto non interessa sapere come il sistema fornisca il comportamento richiesto.

2.2 Class diagram

2.3 Sequence diagram

È un diagramma di iterazione

- Sono diagrammi di comportamento che modellano le interazioni tra varie entità di un sistema.
- Visualizzano lo scambio di messaggitra entità nel tempo.
- Il loro scopo è mostrare come un certo comportamento viene realizzato dalla collaborazione delle entità in gioco.
- adatto a mostrare la sequenza temporale degli avvenimenti per ogni entità nel diagramma.

2.4 Activity diagram

I diagrammi di attività descrivono un flusso di azioni che realizzano un certo comportamento specifico. L'enfasi non è sullo scambio di messaggi ma sui blocchi di comportamento.
Il diagramma di attività modella un comportamento (che riguarda una o più entità) come un insieme di azioni organizzate secondo un flusso.

2.5 Analisi vs Progettazione

- L'analisi modellai concetti chiave del dominiodel problema.
- La progettazione adattail modello di analisi e lo completaaffinché diventi implementabile.
- L'analisi è vicina al problema.
- La progettazione è vicina alla soluzione.

3 Vario

3.1 Agile Developement

Program specification, design and implementation are inter-leaved
The system is developed as a series of versions or increments with stakeholders involved in version specification and evaluation
Frequent delivery of new versions for evaluation
Minimal documentation – focus on working code
The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project

3.2 Plan-driven development

A plan-driven approach to software engineering is based around separate development stages with the outputs to be produced at each of these stages planned in advance.
Not necessarily waterfall model – plan-driven, incremental development is possible