

1) Il pattern è un modo di risolvere un problema noto
Determinando una soluzione riutilizzabile.

2) Una variabile di un certo tipo T
pos. contenere tipi T e sottotipi di T
es: Animale a = new Dog();

3) Nello sviluppo OO è importante utilizzare Dipendenza.

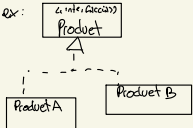
Tra Interfacce e Classi:

(perché?)

perché ci fornisce RIGIDITÀ

Riutilizzo del Codice

es:



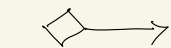
4) Nella Soluzione del Mediator

Una sottoclasse di Mediator chiama i metodi di alcune sottoclassi.

5) Nel Design Pattern, i client conoscono


conoscono la classe Context perciò nessuna
delle classi di State

6) Le relazioni di Aggregazione

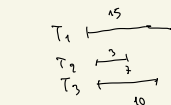


7) Diagrammi Attività Rappresenta

Attività  Rettangoli Arrotondati

Flussi  Freccia

8) $T_1 = 15g$ $T_2 = 3g$ $T_3 = 7g$



9) QUANDO SI DOVRÀ LAVORARE A STRETTO Contatto col Cliente
Spirale, XP

tanti processi di sviluppo perciò maggiore
contatto col Cliente.

10) Il processo XP produce

2) Poca Documentazione

11) Un metodo di tipo SYNCHRONIZED

Permette l'esecuzione di un solo Thread all'interno dell'oggetto

12) Un'interfaccia X, ovvero public interface X

Ha lo scopo di definire un tipo

Inoltre:
Permette alle classi che la usano di non legarsi ad una implementazione

13)

Leggi di Lehman:

- I sistemi dovrebbero essere CAMBIATI per essere UTILI.

14)

```
public Libro getLibro() { return b.getLibro(); }
```

- è compatibile solo se getLib() restituisce un'istanza di Libro

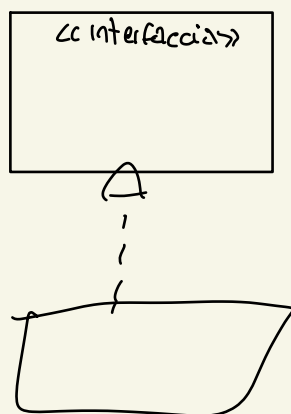
15) I requisiti utente sono più spesso descritti tramite
Linguaggio Formale, Tabelle ecc

16) il metodo newInstance() è fornito dalla classe
Class

17) Nel designi Pattern Observer si desidera ottenere

Indipendenze tra oggetto ed il numero e il tipo di Osservatori

18) Interfaccia Come viene implementata?



19) Class Adapter

L'Adapter eredita Adaptee.

Inoltre Elimina l'uso di una istanza all'interno di Adapter.

20) Nel Diagramma di Attività vi sono
Tabelle, Diagrammi di Attività, Reti di Attività

21) È Consigliabile usare il Facade
Quando si vuole nascondere la complessità di un sottosistema.

22) Un documento dei Requisiti ben scritto mirerà a fornire
Fornisce Informazioni da cui derivare la progettazione

23) Nella soluzione del design State, ogni Concrete State
Implementata in COMPORTAMENTO.

24) Observer cosa è interessante Osservare
Lo stato del Concrete Subject

25) La coesione di una classe è
Alta se tutti i metodi contribuiscono ad
implementare un singolo compito.

26) La legge di Lehman è pensato per
Software di grandi Dimensioni.

27) **DOMANDE RISPOSTA Aperta:**
Disegnare L'ANALISI DEI REQUISITI
• Studio Fattibilità.

• Analisi dei Requisiti.

• Specifiche dei Requisiti.

• Convalida dei Requisiti.

CODICE

```

public interface Auto {
    public String getTipo();
    public int getPeso();
    public float getDistanza(int t);
}

public interface Motore {
    public int getPotenza();
}

public class Berlino implements Auto {
    private Motore m;
    public Berlino(Motore x) { m = x; }
    public String getTipo() { return "Berlina"; }
    public int getPeso() { return 800; }
    public float getDistanza(int t) {
        return (float) t * t * m.getPotenza() / getPeso();
    }
}

public class Fire implements Motore {
    public int getPotenza() { return 1000; }
}
    
```

