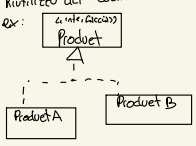
1) I pattern Permettono di risolvere un problema noto Descrivendo una soluzione riutilizzabile

2) Una variabile di un certo tipo T può contenere tipi T e Sottotipi di T
ex: Animale a = new Dog();

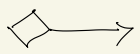
3) Nello sviluppo OO è incoraggiato utilizzare Dipendenze TRA Interfacce e Classi:
(perché?)
perché ci fornisce MODULARITÀ
Riutilizzo del Codice
ex:



4) Nella Soluzione del Mediator
Una sottoclasse di Mediator chiama i metodi di alcune sottoclassi.

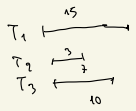
5) Nel Design Pattern, i client Conosce
Conosce la classe Context perciò nessuna delle sottoclassi di State

6) Le relazioni di Aggregazione



7) DIAGRAMMA Attività Rappresenta
Attività  Rettangoli Arrotondati
Flussi ⟶ Freccia

8) $T_1 = 5g$ $T_2 = 3g$ $T_3 = 7g$



9) QUANDO si DOVRÀ LAVORARE A STRETTO Contatto col Cliente
Spirale, XP
tanti processi di sviluppo perciò MAGGIOR Contatto col Cliente.

10) Il processo XP produce
2) Poca Documentazione

11) Un metodo di tipo SYNCRONIZED
Permette l'esecuzione di un solo Thread all'interno dell'oggetto

12) Un interfaccia X, avero public interface X
Ha lo scopo di definire un tipo
Inoltre:
Permette alle classi che la usano di non legarsi ad un implementazione

13)
Leggi di Lehman:
- I sistemi dovrebbero essere CAMBIATI per essere UTILI.

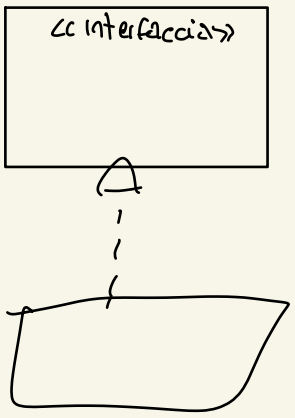
14)
publie Libro getLibro() {return b.getLib();}
- è compatibile solo se getLib() restituisce un' istanza di LiBro

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

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

17) Nel design 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 ConcreteState
Implementa un COMPORTAMENTO.

24) Observer cosa è interessante Osservare
Lo stato del ConcreteSubject

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'ANACISI DEI REQUISITI
- Studio Fattibilità.
- Analisi dei Requisiti.
- Specifiche dei Requisiti.
- Convalida dei Requisiti.

28)

CODICE

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

public interface Motore {
  public int getPotenza();
}

public class Berlina implements Auto {
  private Motore m;
  public Berlina(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; }
}
```