Esercitazione Capitolo 4:

Java

1. Binding dinamico in Java

Date le seguenti classi:

```
class L{ void print(double l){System.out.println("L");}}
class M extends L{ void print(int l){System.out.println("M");}}
```

Se faccio:

```
int h = 6;
L l = new M();
l.print(h);
```

Cosa stampo? Individua la segnatura selezionata a compile time (early binding) e quella a runtime (late binding). Come potrei modificare per attivare il binding dinamico se non è già attivato?

2. Equals in Java

Definisci una classe Persona con un campo codice fiscale (16 caratteri). Due Persona sono uguali (equals) se hanno lo stesso codice fiscale. Prova a definire equals con questa segnatura:

public boolean equals(Persona p){....}

Riesci a fare un esempio in cui due Persona pur avendo lo stesso codice fiscale restituiscono equals false?

Come dovresti modificare il metodo equals?

3. Covarianza valore restituito

Dichiara una classe B sottoclasse di A che ridefinisce un metodo cambiandone il tipo del valore ritornato con un sottotipo di quello originale definito in A. Controlla che effettivamente se runtime l'oggetto è una istanza di B, il valore ritornato ha tipo corretto.

4. Covarianza degli array

Covariance

if S <: T then S[ ] <: T[ ]

S <:T means “S is subtype of T”

Dichiara un array S[] s e un T[] t. Puoi assegnare t = s ? Inserisci ora un T in t ... controlla che avrai eccezione ...

5. Generics in Java SequenzaOrdinabile

Definisci una struttura dati SequenzaOrdinabile che è generica, di elementi ordinabili. Ha sottostante un array (puro di Object). Ha i seguenti metodi:

- **insert** che inserisce nella sequenza
- metodo **sort** che ordina (usando ad esempio bubble sort)
- **toString**

Prova a fare un main in cui provi con le stringhe.

Se hai una classe Person (con unico campo nome) che è ordinabile rispetto al nome e una classe Student che è sottoclasse di Person, riesci ad usare SequenzaOrdinabile<Student> ?

Prova definire in una classe NON generica SequenzaUtils un metodo statico che prende una SequenzaOrdinabile e restituisce l'elemento di mezzo (mean).

6. Generics in Java - glossario

Scrivi una classe generica che rappresenta un dizionario come una lista di coppie di tipo A e B (definisci la classe Coppia) in cui il primo elemento A è ordinabile. Questa classe ha un metodo **insertInOrdine** che inserisce una coppia in ordine rispetto il primo elemento della coppia.
Ha un metodo **find** che cerca un A e se lo trova (il primo che trova) restituisce il B con cui A è accoppiato. Se non c'è restituisce null.
Ha un metodo toString che stampa il glossario in ordine.
Scrivi un main in cui fai un po' di prove.

7. Generics in Java + Visitor

Un componente può essere composto da componenti o essere atomico. Voglio calcolare il prezzo di un componete e ottenere una stringa (nome) che lo rappresenti.

- **Prezzo** (int) – è 10 se è atomico o pari alla somma del prezzo dei componenti.
- **Nome** (string) – 'C' se atomico, '[' nome1, … nome_n ']' se composto dove nome_i è il nome dell'i-esimo componente.

Usa il visitor pattern (generico sul tipo restituito meglio) per implementare le operazioni richieste.
Scrivi nella stessa classe un metodo static **sommaCosto** che data una Lista di componenti calcola la somma dei costi utilizzando il visitor e la restituisce. Definisci **sommaCosto** in modo che possa prendere sia lista di componenti atomiche o composte.

Fai un main con un esempio in cui creai almeno 4 componenti atomiche e 4 composti che li contengano, crei i visitor e calcoli sia il costo che il nome con il visitor, crea alcune liste e chiama sommaCosto.

8. Facade e singleton

Scrivi un esempio di pattern facade e sigleton in Java. Ad esempio Terra è un sigleton che ha come componenti diversi parti della terra (crosta, nucleo, atmosfera....). Terra fa anche da Facade (ad esempio per dimensione della crosta, etc.)

9. Varargs

Scrivi un metodo print che prende un numero variabile di stringhe e le stampa una dopo l'altra

10. Maps

Scrivi una piccola applicazione in cui leggi un file di parole (o dallo standard input) e poi alla fine stampi la frequenza con cui queste appaiono nel file. Puoi usare Scanner e il suo metodo next per leggere una parola alla volta da file.

11. Collections uso avanzato

Scrivi un piccolo programmino in cui

- usi almeno quattro classi del Java Collection Framework,
- di cui almeno una è una Map e almeno una è una interfaccia
- estendi una collezione
- almeno una è una sorted
- usi almeno due metodi di classe (algorithms)
- usi almeno due metodi di utilità

Whimsical Toys Inc (WTI) needs to record the names of all its employees. Every month, an employee will be chosen at random from these records to receive a free toy.
WTI has decided that each new product will be named after an employee — but only first names will be used, and each name will be used only once. Prepare a list of unique first names.
WTI decides that it only wants to use the most popular names for its toys. Count the number of employees who have each first name.
WTI acquires season tickets for the local lacrosse team, to be shared by employees. Create a waiting list for this popular sport.