ESAME 2 LUGLIO 2021

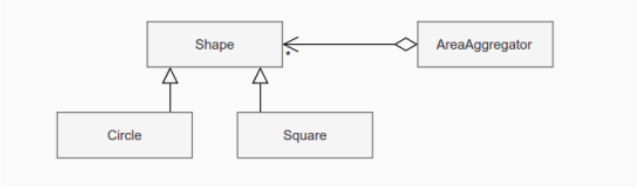
1. Descrivere il concetto di interfaccia, spiegando le motivazioni che stanno alla base del suo utilizzo e i vantaggi che apporta allo sviluppo. Produrre un esempio **minimale** scritto in Java che concretizzi quanto spiegato.

UN INTERFACCIA È UN COSTRUTTO DELLA PROGRAMMAZIONE AD OGGETTI CHE HA LO SCOPO DI DESCRIVERE IL COMPORTAMENTO DI UNO SPECIFICO INSIEME DI CLASSI. OGNI CLASSE PUO' FAR PARTE DI PIU' INTERFACCE TRAMITE LA KEYWORD "IMPLEMENTS".
UN INTERFACCIA SI DEFINISCE IN MODO SIMILE AD UNA CLASSE, ESSA NON HA COSTRUTTORE, ED HA ATTRIBUTI E METODI ASTRATTI CHE LE CLASSI CHE LA IMPLEMENTERANNO DOVRANNO OBBLIGATORIAMENTE SOVRASCRIVERE, PUO', PERO' AVERE UN METODO DI "DEFAULT" GIA' DEFINITO, CONDIVISO FRA TUTTE LE CLASSI CHE LO IMPLEMENTANO. AD ESEMPIO, TUTTE LE CLASSI CHE DESCRIVONO UN INSIEME DI ELEMENTI, POTREBBERO FAR PARTE DI UN IPOTETICA INTERFACCIA "COUNTABLE".

esempio in Java:

```
PUBLIC INTERFACE COUNTABLE {
    INT GETCOUNT();
}
PUBLIC CLASS DOUBLELIST IMPLEMENTS COUNTABLE{
    @OVERRIDE
    PRIVATE ARRAYLIST LISTA = NEW ARRAYLIST();
    PRIVATE ARRAYLIST LISTB = NEW ARRAYLIST();
    PUBLIC INT GETCOUNT(){
        RETURN LISTA.GETSIZE()+LISTB.GETSIZE();
    }
}
```

2. Sia data l'implementazione del seguente diagramma UML, composto da una classe *AreaAggregator* realizzata con lo scopo di calcolare l'area di una qualsiasi classe derivante da *Shape*:



La classe *AreaAggregator* è implementata nel seguente modo:

```
public class AreaAggregator {
        private ArrayList<Shape> shapes = new ArrayList<>();

        public void addShape(Shape shape) {
          shapes.add(shape);
        }
        public double sum() {
          double sum = 0;

          Circle c = new Circle(6);
          Circle c2 = new Circle(3.5);
          this.addShape(c);
          this.addShape(s);
          this.addShape(c2);

          for (Shape shape: shapes) {
            if(shape.getName() == "Circle") {
              int radius = ((Circle) shape).getRadius();
              sum += Math.PI * Math.pow(radius, 2);
            } else if (shape.getName() == "Square") {
              double side = ((Square) shape).getSide();
              sum += Math.pow(side, 2);
            }
            counter+= 1;
          }

          System.out.println("Sum:" + sum());
          return String.valueOf(sum); // Converto il risultato in stringa
        }
      }
```

← S NON ESISTE

NON DOVREBBE ESSERE NELLA FUNZIONE CHE FA LA SOMMA (SINGLE PURPOSE PRINCIPLE)

← SI DOVREBBE RIFERIRE ALLA CLASSE

← IL RAGGIO NON DOVREBBE ESSERE DI TIPO INT.

← LA LIBRERIA MATH NON È IMPORTATA

← SI USA UN FOR, COUNTER NON SERVE E NEMMENO ESISTE.

← STA CHIAMANDO LA FUNZIONE SUM() (RICORSIONE INFINITA), ED È SBAGLIATO ESEGUIRE UN PRINT DA QUI.

← LA FUNZIONE NON DOVREBBE RITORNARE UNA STRINGA

Come si può notare, questa implementazione presenta alcuni errori di tipo sintattico, semantico ed un approccio *object oriented* non del tutto idoneo. Elencare quindi le anomalie e motivarle in modo opportuno. È consentito l'uso di codice Java.

NB: si assume che i metodi *get* siano *public* e che forniscano sempre l'attributo atteso.

3. Sia dato un file contenente le informazioni relative agli studenti iscritti ad un corso. Tali informazioni comprendono nell'ordine: la matricola, il nome, il cognome ed il voto verbalizzato, se presente. I campi sono separati dal carattere ":". Un esempio di contenuto del file è il seguente:

1:Michele:Apicella:
2:Salazar:Slytherin:24
65:Neville:Longbottom:31
74:Giulio:Ponte:25
...

Si realizzi un programma che, leggendo da input questo file, provveda a:

- Mostrare la lista degli studenti che hanno già verbalizzato;
- Mostrare la lista degli studenti che devono ancora verbalizzare;
- Mostrare le statistiche del corso (studenti iscritti, studenti che hanno superato l'esame, voto medio verbalizzato)

```
PUBLIC   CLASS  STUDENT {
        PRIVATE  INT  MAT;
        PRIVATE STRING NAME;
        PRIVATE STRING SURNAME;
        PRIVATE INT  GRADE;
        PUBLIC  STUDENT(STRING  LINE){
                STRING DATA =  LINE.SPLIT(":");
                MAT = INTEGER.PARSEINT(DATA[0]);
                NAME =  DATA[1];
                SURNAME = DATA[2];
                IF(DATA.LENGHT() >3) {
                        GRADE=DATA[4];
                }ELSE { GRADE = -1;}
        }
        PUBLIC INT  GETMAT(){  RETURN  MAT;}
        PUBLIC STRING  GETNAME  (){RETURN  NAME ;}
        PUBLIC STRING  GETSURNAME(){RETURN SURNAME;}
        PUBLIC INT  GETGRADE() { RETURN  GRADE;}
}
IMPORT  JAVA.UTIL.*;
IMPORT  JAVA.IO.*;
PUBLIC   CLASS  MAIN {
        PUBLIC  STATIC  VOID  MAIN(STRING[ ] ARGS) {
                FILE  F :NEW  FILE(" PERCORSO/ FILE.TXT");
                ARRAYLIST < STUDENT> STUDENTS = NEW  ARRAYLIST<>();
                TRY{
                        SCANNER  IN  = NEW  SCANNER(F);
                        WHILE(IN.HASNEXT()){STUDENTS.ADD(IN.NEXTLINE());}
                } CATCH(FILENOTFOUNDEXCEPTON  E) {
                        SYSTEM.OUT.PRINT(" ERRORE  FILE");
                        RETURN;
                }
                INT  DONE  = 0;
                INT  PASSED =0;
                INT  GRADESUM = 0;
                SYSTEM.OUT.PRINTLN (" GLI  STUDENTI  ISCRITTI  SONO : " + STRING.VALUEOF( STUDENTS.SIZE()));
                FOR(INT i=0;i<STUDENTS.SIZE(); i++){
                        INT  TMPGRADE =(STUDENT)STUDENTS. GET(i). GETGRADE();
                        STRING  NAME =(STUDENT)STUDENTS. GET(i).GETNAME+"  "+(STUDENT)STUDENTS. GET(i).GETSURNAME();
                        IF(TMPGRADE == -1) {
                                SYSTEM.OUT.PRINTLN(NAME+"  NON HA  VERBALIZZATO");
                        ELSE  {
                                SYSTEM.OUT.PRINTLN(NAME+"  HA  VERBALIZZATO);
                                GRADESUM += GRADE;
                                DONE += 1;
                                IF  (GRADE >17){ PASSED +=1;}
                        }
                }
                SYSTEM.OUT. PRINTLN (" MEDIA  :"+ STRING.VALUEOF(GRADESUM/DONE)+" PASSATI :" + PASSED);
        }
}
```

4. Una rubrica telefonica offre la possibilità di memorizzare uno o più contatti, ciascuno di essi rappresentato da un nome, un cognome ed un numero telefonico. Realizzare un programma che permetta di:

- Restituire la rappresentazione testuale dell'intera rubrica;
- Aggiungere un contatto alla rubrica;
- Rimuovere dalla rubrica il contatto associato al nome e cognome forniti in input;
- Cercare il numero di telefono associato ad un nome e cognome forniti in input;

Successivamente, scrivere un programma di test che utilizzi tutte le funzionalità implementate.

```
PUBLIC CLASS CONTACT {
        STRING NAME;
        STRING SURNAME;
        INT NUM;
        PUBLIC CONTACT(STRING N, STRING S, INT NUMBER){
                NAME=N;
                SURNAME = S;
                NUM=NUMBER;
        }
        PUBLIC STRING GETNAME(){ RETURN NAME;}
        PUBLIC STRING GETSURNAME(){ RETURN SURNAME;}
        PUBLIC INT GETNUM(){ RETURN NUM;}
}
IMPORT JAVA.UTIL.*;
PUBLIC CLASS PHONEBOOK {
        PRIVATE ARRAYLIST<CONTACT> PHONELIST = NEW ARRAYLIST<>();
        PUBLIC VOID PRINTCONTACTS(){
                FOR(INT i=0;i<PHONELIST.SIZE();i++){
                        CONTACT TMP=(CONTACT) PHONELIST.GET(i);
                        SYSTEM.OUT.PRINTLN(TMP.GETNAME()+" "+TMP.GETSURNAME()+" "+STRING.VALUEOF(TMP.GETNUM()));
                }
        }
        PUBLIC VOID ADDCONTACT(STRING N, STRING S, INT K){
                PHONELIST.ADD(NEW CONTACT(N,S,K));
        }
        PRIVATE VOID SEARCHINDEX(STRING NAME, STRING SURNAME){
                FOR(INT i=0;i<PHONELIST.SIZE();i++){
                        CONTACT TMP=(CONTACT) PHONELIST.GET(i);
                        IF(TMP.GETNAME()==NAME && TMP.GETSURNAME()==SURNAME){ RETURN i;}
                }
                RETURN -1;
        }
        PUBLIC VOID REMOVE(STRING NAME, STRING SURNAME){
                INT i=SEARCHINDEX(NAME,SURNAME);
                IF(i!=-1){PHONELIST.REMOVE(i);} ELSE { SYSTEM.OUT.PRINTLN("NON TROVATO");}
        }
        PUBLIC VOID SEARCH(STRING NAME, STRING SURNAME){
                INT i=SEARCHINDEX(NAME,SURNAME);
                IF(i!=-1){SYSTEM.OUT.PRINTLN(STRING.VALUEOF((CONTACT) PHONELIST.GET(i).GETNUM));}
                ELSE { SYSTEM.OUT.PRINTLN("NESSUN NUMERO");}
        }
```