POLITECNICO
MILANO 1863

Ingegneria del Software

Ingegneria Informatica – Sez. P-Z

Esercitazione:

Java

continua dalla scorsa esercitazione

– esercizio introduttivo – Poligoni

Si scriva una classe che rappresenti un poligono nell'(iper)piano.

Ogni poligono è costituito da una successione di punti, ognuno descritto tramite le sue coordinate cartesiane.

Per ogni poligono deve essere possibile calcolare il perimetro.

Una volta creato, i vertici del poligono possono essere spostati e il perimetro deve essere calcolato solo una volta per ogni modifica.

– esercizio introduttivo – Poligoni

Si scriva una classe che rappresenti un poligono nell'(iper)piano

Prima di procedere …

vediamo come possiamo implementare il metodo di PuntoND «distanzaDa(Punto p)».

Una volta creati, i vertici del poligono possono essere spostati e il perimetro deve essere calcolato solo una volta per ogni modifica.

– esercizio introduttivo –
Poligoni «punto per punto»

Si scriva una classe che rappresenti un poligono nell'(iper)piano.

Ogni poligono è costituito da una successione di punti, ognuno descritto tramite le sue coordinate cartesiane.

Per ogni poligono deve essere possibile calcolare il perimetro.

Una volta creato, i vertici del poligono possono essere spostati e il perimetro deve essere calcolato solo una volta per ogni modifica.

Deve essere possibile costruire un Poligono «un punto alla volta» partendo da un Poligono senza vertici. Un Poligono senza vertici non ha perimetro, mentre un Poligono non ancora «chiuso» ha un perimetro che non include il lato che parte dall'ultimo vertice.

– esercizio introduttivo –
Poligoni «punto per punto»

Si scriva una classe che rappresenti un poligono nell'(iper)piano.

Ogni poligono è costituito da una successione di punti, ognuno descritto tramite le sue coordinate cartesiane.

Per ogni poligono deve essere possibile calcolare il perimetro.

Una volta creato, i vertici del poligono possono essere spostati e il perimetro deve essere calcolato solo una volta per ogni modifica.

Deve essere possibile costruire un Poligono «un punto o gruppo di punti alla volta» partendo da un Poligono senza vertici. Un Poligono senza vertici non ha perimetro, mentre un Poligono non ancora «chiuso» ha un perimetro che non include il lato che parte dall'ultimo vertice.

– esercizio introduttivo –
Poligoni «punto per punto» propri

Si scriva una classe che rappresenti un poligono nell'(iper)piano.

Ogni poligono è costituito da una successione di punti (ciascuno differente da tutti gli altri), ognuno descritto tramite le sue coordinate cartesiane.

Per ogni poligono deve essere possibile calcolare il perimetro.

Una volta creato, i vertici del poligono possono essere spostati e il perimetro deve essere calcolato solo una volta per ogni modifica.

Deve essere possibile costruire un Poligono «un punto o gruppo di punti alla volta» partendo da un Poligono senza vertici. Un Poligono senza vertici non ha perimetro, mentre un Poligono non ancora «chiuso» ha un perimetro che non include il lato che parte dall'ultimo vertice.

– esercizio introduttivo – Forme

Un Poligono non è altro che una Forma.

Ogni Forma è caratterizzata da una superficie colorata.

Per ogni Forma deve essere possibile calcolarne il perimetro.

Esercizi

Esercizio 1 - TdE 26/6/2020 (es3)

Si consideri il seguente programma Java

```
abstract class Person {
        public void join(Lecture lec) {
                System.out.println(”Joining ” + lec); }
        public void join(OnlineLecture lec) {
                System.out.println(”Joining ” + lec); }
}
class Student extends Person {
        public void join(Lecture lec) {
                System.out.println(”Student joining ” + lec); }
}
class Teacher extends Person {
        public void join(OnlineLecture lec) {
                System.out.println(”Teacher joining ” + lec); }
}
class Lecture {
        public void addAttendant(Person p) { p.join(this); }
        public String toString() { return ”a lecture”; }
}
class OnlineLecture extends Lecture {
        public String toString() { return ”an online lecture”; }
}
```

Esercizio 1 - TdE 26/6/2020 (es3)

(… continua …)

```
 public class Main {
      public static void main( String [] args ) {
1           Person p1 = new Student();
2           Person p2 = new Person();
3           Person p3 = new Teacher();
4           Student p4 = new Student();
5           Student p5 = new Teacher();
6           Lecture lec1 = new Lecture();
7           Lecture lec2 = new OnlineLecture();
8           OnlineLecture lec3 = new OnlineLecture();
                              […]
```

Esercizio 1 - TdE 26/6/2020 (es3)

```
public class Main {
        public static void main( String [] args ) {
                                        […]
9               lec1.addAttendant(p1);
10              lec1.addAttendant(p2);
11              lec1.addAttendant(p3);
12              lec1.addAttendant(p4);
13              lec1.addAttendant(p5);
14              lec2.addAttendant(p1);
15              lec2.addAttendant(p2);
16              lec2.addAttendant(p3);
17              lec2.addAttendant(p4);
18              lec2.addAttendant(p5);
19              lec3.addAttendant(p1);
20              lec3.addAttendant(p2);
21              lec3.addAttendant(p3);
22              lec3.addAttendant(p4);
23              lec3.addAttendant(p5);
}}
```

Esercizio 1 - TdE 26/6/2020 (es3)

Quesiti:

a) Quali linee del programma non sono valide? Motivare brevemente la risposta.

b) Dopo aver rimosso tutte le linee non valide identificate al punto precedente, indicare cosa stampa il programma, motivando brevemente la risposta.

continua dalla scorsa esercitazione

Esercizio 2 - PointStackedSet

Implementare una classe PointStackedSet:

- Che raccolga un numero arbitrario di Point (punti nello spazio bidimensionale), tutti differenti fra loro.
- *(push)* Che permetta di aggiungere alla raccolta un Point, mantenendo\aggiornando l'ordine di inserimento
- *(pop)* Che permetta di rimuovere e restituire dalla raccolta un Point, nell'ordine inverso rispetto a quello di inserimento.

continua dalla scorsa esercitazione

Esercizio 2 - PointStackedSet

Si estendano le funzionalità precedenti in modo da permettere:

- Di creare un nuovo PointStackedSet come copia di un PointStackedSet esistente
- Di aggiungere alla raccolta più Point contemporaneamente
- Di rimuovere e restituire dalla raccolta più Point contemporaneamente

Esercizio 3 - Pokèmon

Un Pokèmon ha un nome, un tipo (normale, erba, fuoco, acqua), un fattore di forza e dei punti vita.

Un Pokèmon può attaccare un altro Pokèmon, in tal caso, gli procura un danno pari al fattore di forza, eventualmente aumentato del 20% in caso di superefficacia.

Un attacco è superefficace in base al confronto dei tipi di attaccante e avversario:
(erba → acqua, acqua → fuoco, fuoco → erba)

Ditto è un Pokèmon particolare di tipo normale. Quando attacca un Pokèmon, ne acquisisce il tipo. Questa «trasformazione» rimane attiva fino a che i suoi punti vita non scendono sotto i 10. Quando ciò succede, Ditto ritorna di tipo normale.