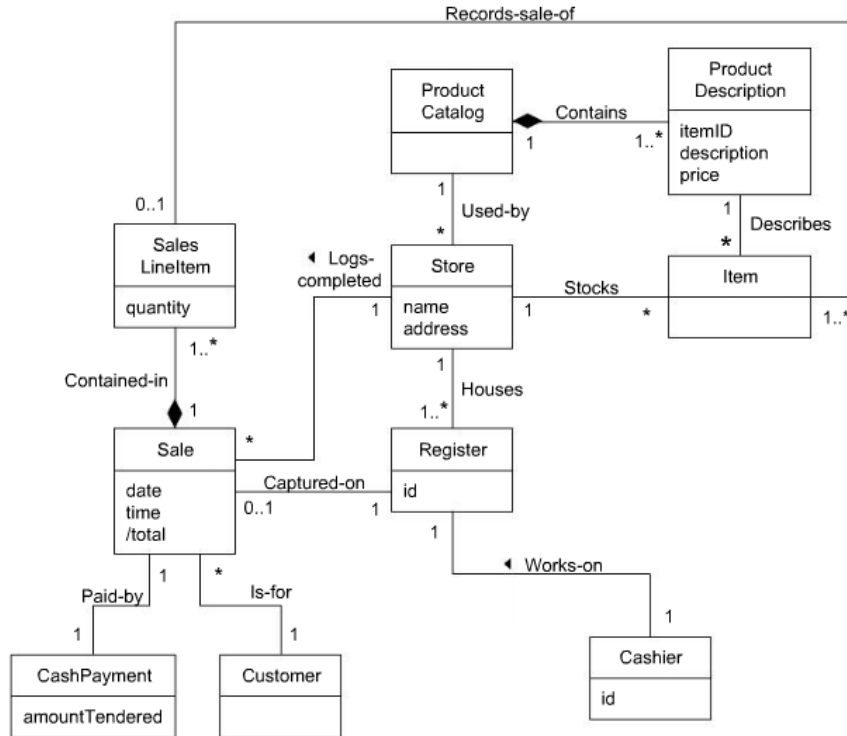


Esercizi in classe di Sviluppo di Applicazioni Software

Matteo Baldoni

May 9, 2022

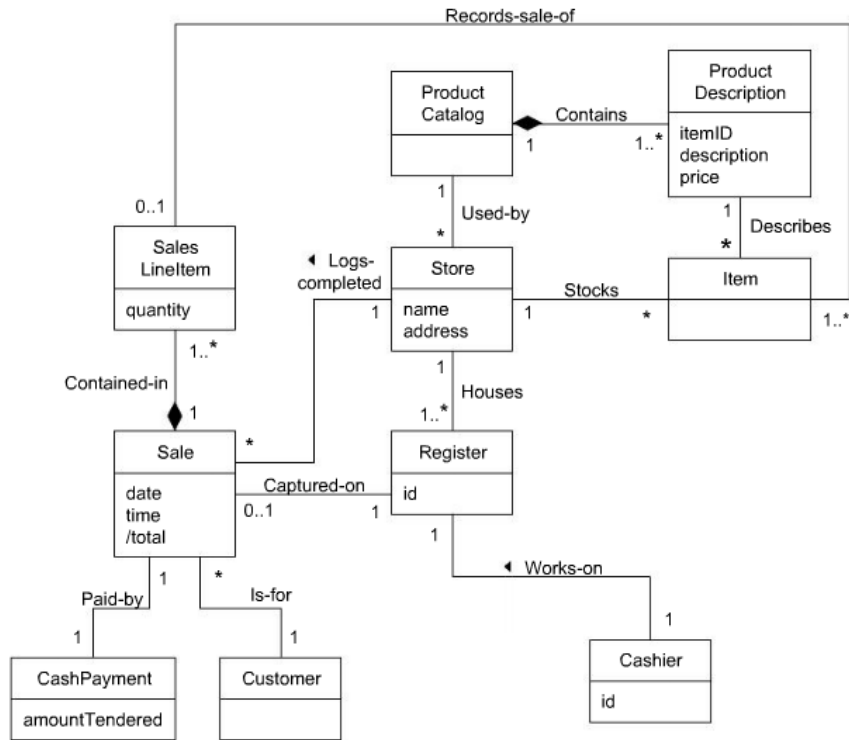
1. Si consideri l'associazione "Contained-in" nel seguente *Modello di Dominio*:



Dire quali delle seguenti affermazioni sono vere e quali false.

- ☒ Una istanza di *Sale* deve essere distrutta prima di un'istanza di *SalesLinItem*.
- ☒ Una istanza di *SalesLinItem* deve essere distrutta prima di un'istanza di *Sale*.
- ☒ Le istanze di *SalesLinItem* appartengono ad una sola istanza di *Sale* alla volta.
- ☒ Le istanze di *Sale* appartengono ad una sola istanza di *SalesLinItem* alla volta.
- ☒ Una istanza di *SalesLinItem* può essere creata dopo un'istanza di *Sale*.
- ☐ Una istanza di *Sale* può essere creata dopo un'istanza di *SalesLinItem*.

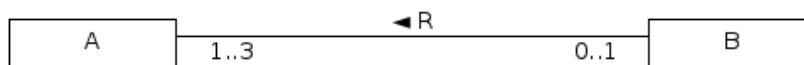
2. Si consideri il seguente *Modello di Dominio*:



Si supponga $Product\ Catalog = \{c_1, c_2\}$ e $Product\ Description = \{d_1, d_2, d_3, d_4\}$. Dire quali delle seguenti affermazioni sono vere e quali false.

- ☒ $Contains$ può essere $\{(c_1, d_1), (c_1, d_2), (c_2, d_2), (c_2, d_3), (c_2, d_4)\}$.
- ☒ $Contains$ può essere $\{(c_1, d_1), (c_1, d_2), (c_1, d_3), (c_2, d_4)\}$.
- ☒ $Contains$ può essere $\{(c_1, d_1), (c_1, d_2), (c_2, d_3)\}$.
- ☒ $Contains$ può essere $\{(c_1, d_1), (c_1, d_2), (c_1, d_3), (c_1, d_4)\}$.

3. Si consideri il diagramma seguente:



Si supponga $A = \{a_1, a_2, a_3, a_4\}$ e $B = \{b_1, b_2\}$.

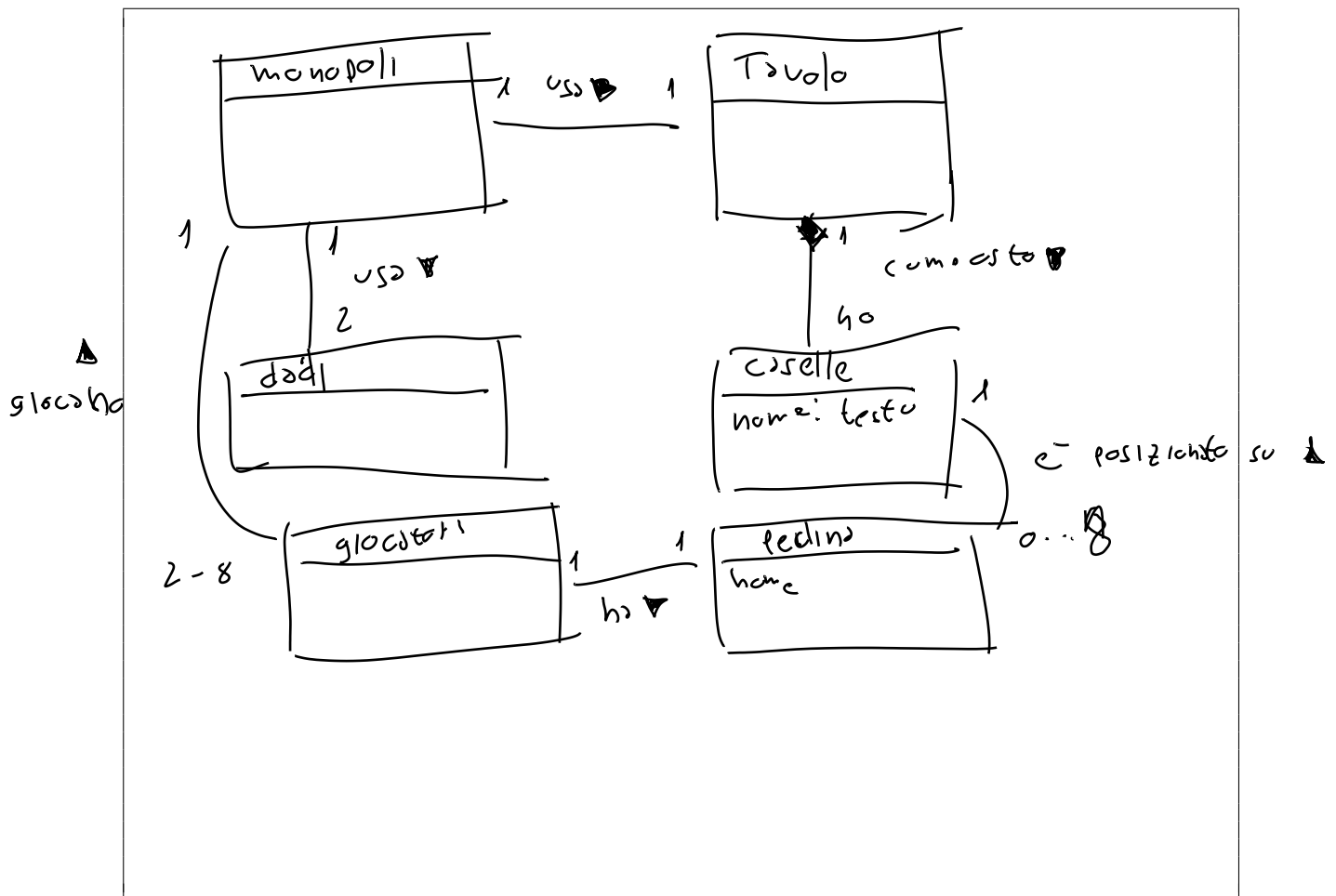
Dire quali delle seguenti affermazioni sono vere e quali false.

- ☒ R può essere $\{(b_1, a_1), (b_1, a_2), (b_2, a_1)\}$.
- ☒ R può essere $\{(b_1, a_1), (b_1, a_2), (b_1, a_3)\}$.
- ☒ R può essere $\{(b_1, a_1), (b_1, a_2), (b_1, a_3), (b_1, a_4), (b_2, a_3)\}$.
- ☒ R può essere $\{(b_1, a_1), (b_1, a_2), (b_2, a_3), (b_2, a_4)\}$.

4. Si consideri la seguente descrizione del gioco Monopoli:

- il gioco di Monopoli è giocato su una tavola di gioco specifica per il Monopoli, su di una tavola di gioco si può giocare solo un gioco di Monopoli alla volta;
- la tavola da gioco di Monopoli è costituita da 40 caselle e ogni casella ha il suo nome;
- un gioco di Monopoli utilizza due dadi;
- un dado è utilizzato da un solo gioco di Monopoli alla volta;
- ad un gioco di Monopoli possono giocare da due a otto giocatori;
- un giocatore può giocare un solo gioco di Monopoli alla volta;
- ad ogni giocatore è assegnata una pedina diversa;
- ogni pedina ha un nome;
- ogni pedina è su di una sola casella alla volta;
- una casella può ospitare da zero a 8 pedine.

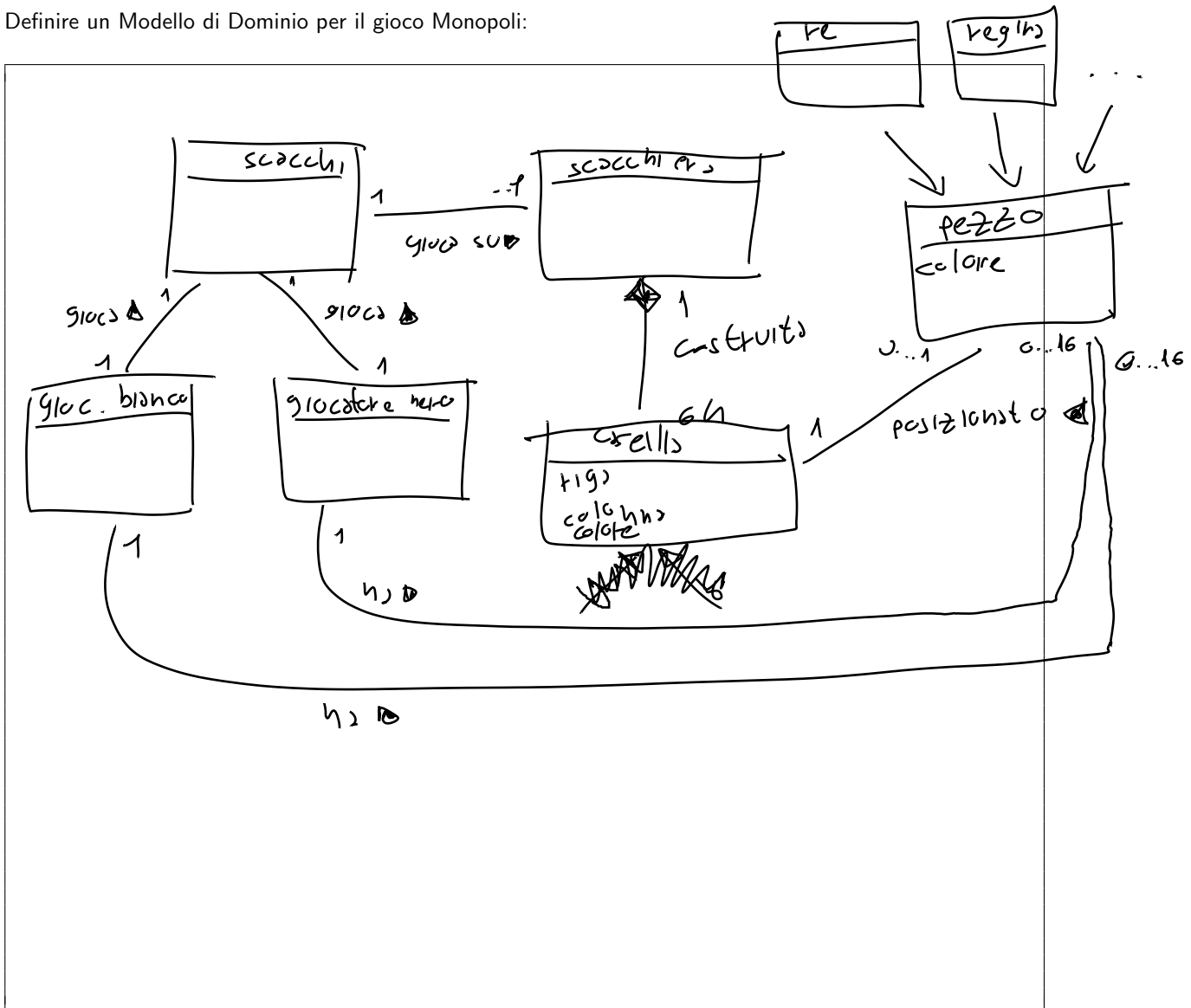
Definire un Modello di Dominio per il gioco Monopoli:



5. Si consideri la seguente descrizione del gioco degli Scacchi:

- il gioco degli scacchi è giocato su una scacchiera, su di una scacchiera si può giocare solo un gioco degli scacchi alla volta;
- la scacchiera è costituita da 64 caselle e ogni casella è individuata da una lettera che va da *a* a *h* che identifica la colonna e un numero da 1 a 8 che identifica la riga;
- una casella appartiene ad una solo scacchiera e ha un colore che può essere *bianco* o *nero*;
- ad un gioco degli scacchi è giocato da un giocatore, con il ruolo del *bianco* e da un giocatore con il ruolo del *nero*
- in una casella può essere sistemato un solo pezzo tra un *re*, una *regina*, un *alfiere*, una *torre*, un *cavallo* o un *pedone*;
- ogni pezzo deve essere sistemato su una casella;
- un giocatore ha almeno un pezzo degli scacchi e un massimo di 16 pezzi degli scacchi;
- ogni pezzo ha un colore, bianco o nero.

Definire un Modello di Dominio per il gioco Monopoli:



6. La classe DistributoreAutomatico rappresenta un distributore automatico che distribuisce bevande (un solo tipo) dopo l'inserimento del corrispettivo costo in monete.

Le tre operazioni di cui è dotato il distributore, inserisciMoneta, annulla e premiBottone hanno un funzionamento che dipende dallo stato del distributore:

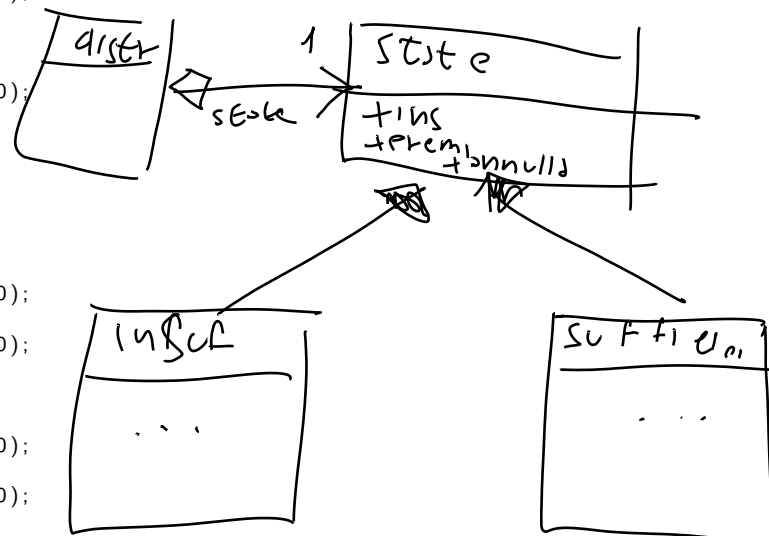
- Nel caso in cui il totale inserito sino a quel momento non corrisponde al prezzo della bevanda, inserisciMoneta aggiorna il totale inserito sino a quel momento mentre premiBottone stampa il costo mancante al raggiungimento del prezzo della bevanda. annulla stampa che il totale inserito fino a quel momento viene azzerato e restituito.
- Nel caso in cui il totale inserito sino a quel momento eguagli o superi il prezzo della bevanda, inserisciMoneta stampa che viene restituito il corrispettivo appena inserito mentre premiBottone stampa che la bevanda è consegnata, il resto restituito e azzer il totale inserito sino a quel momento. annulla stampa che il totale inserito fino a quel momento viene azzerato e restituito.

Si veda il seguente test.

```

1  public class TestDistributoreAutomatico {
2      public static void main(String[] args) {
3          DistributoreAutomatico distributore = new DistributoreAutomatico();
4          System.out.println("--1--");
5          distributore.annulla();
6          System.out.println("--2--");
7          distributore.inserisciMoneta(20);
8          System.out.println("--3--");
9          distributore.premiBottone();
10         System.out.println("--4--");
11         distributore.inserisciMoneta(30);
12         System.out.println("--5--");
13         distributore.premiBottone();
14         System.out.println("--6--");
15         distributore.annulla();
16         System.out.println("--7--");
17         distributore.premiBottone();
18         System.out.println("--8--");
19         distributore.inserisciMoneta(50);
20         System.out.println("--9--");
21         distributore.inserisciMoneta(40);
22         System.out.println("--10--");
23         distributore.premiBottone();
24         System.out.println("--11--");
25         distributore.inserisciMoneta(20);
26         System.out.println("--12--");
27         distributore.inserisciMoneta(50);
28         System.out.println("--13--");
29         distributore.premiBottone();
30     }
31 }

```



Con risultato:

```

--1--
Restituzione di 0
--2--
Totale: 20
--3--
Mancano ancora 80
--4--
Totale: 50
--5--
Mancano ancora 50

```

```
--6--
Restituzione di 50
--7--
Mancano ancora 100
--8--
Totale: 50
--9--
Totale: 90
--10--
Mancano ancora 10
--11--
Totale: 110
Totale raggiunto!
--12--
Restituzione di 50
--13--
Erogazione bevanda
Restituzione resto di 10
```

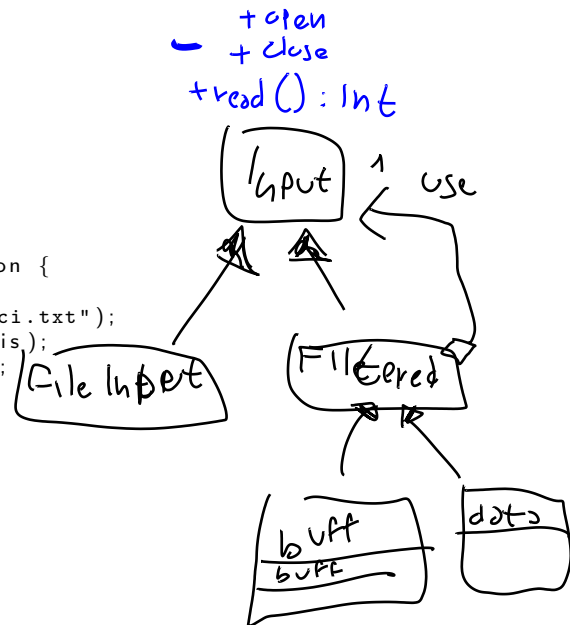
Presentare un Modello di Progetto (diagramma UML delle classi) che utilizzando il pattern GoF *state* realizzi il comportamento descritto.

7. Si consideri il seguente frammento di codice in Java:

```

1 import java.io.*;
2
3 public class ProvaFile {
4     public static void main(String arg[]) throws Exception {
5
6         InputStream is = new FileInputStream("DatiNumerici.txt");
7         FilterInputStream bis = new BufferedInputStream(is);
8         FilterInputStream dis = new DataInputStream(bis);
9
10        int i;
11        char c;
12        try {
13            while ((i = dis.read()) != -1) {
14                c = (char) i;
15                System.out.print(c);
16            }
17        } catch (Exception e) {
18            e.printStackTrace();
19        } finally {
20            if (dis != null) dis.close();
21        }
22    }
23 }

```



Il file DatiNumerici.txt è il seguente:

```

9
13
7

```

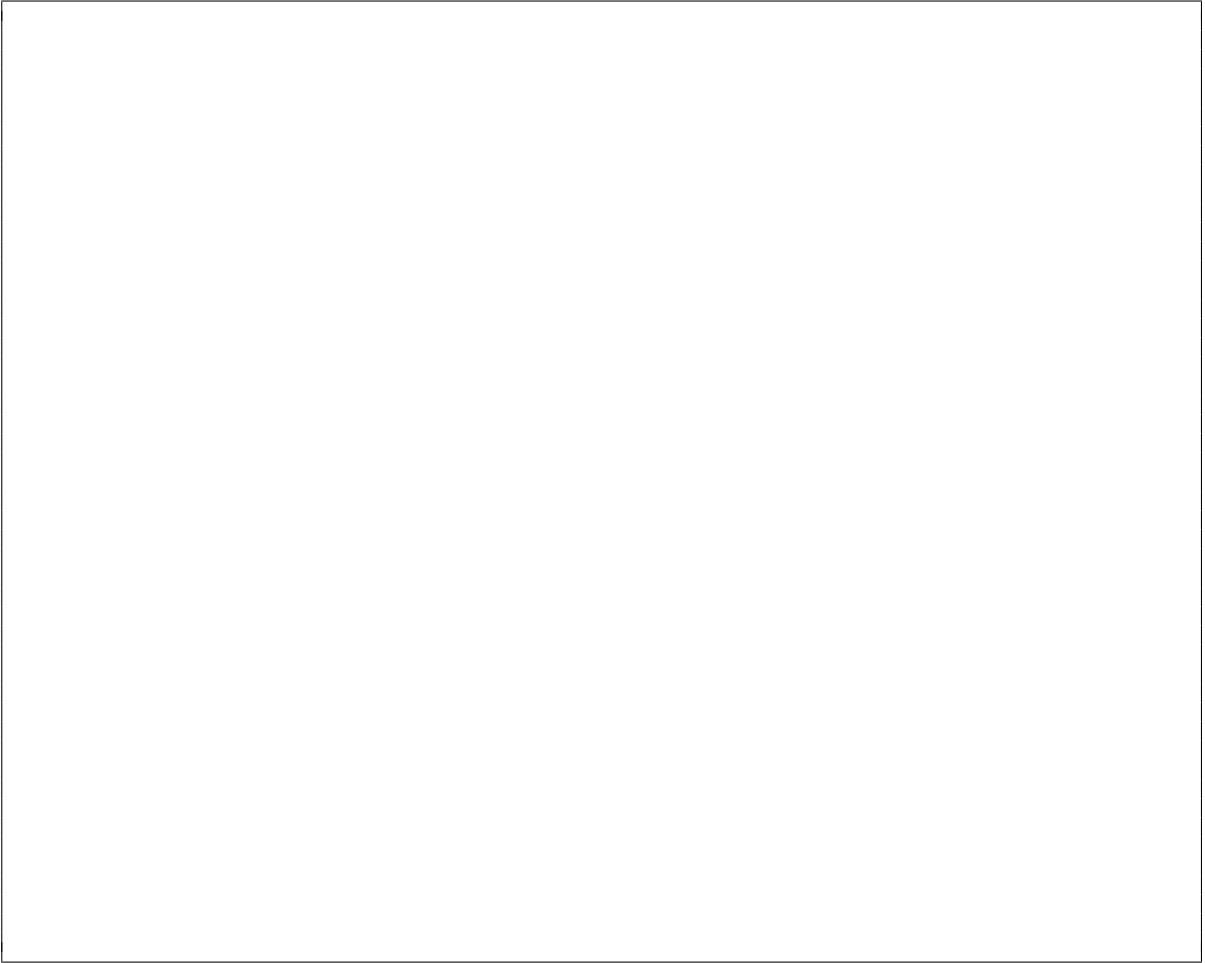
l'ecuzione del main di ProvaFile restituisce il seguente risultato a console:

```

9
13
7

```

Presentare un modello di progetto (diagramma UML delle classi) che utilizzando il pattern decorator realizzi il comportamento descritto.



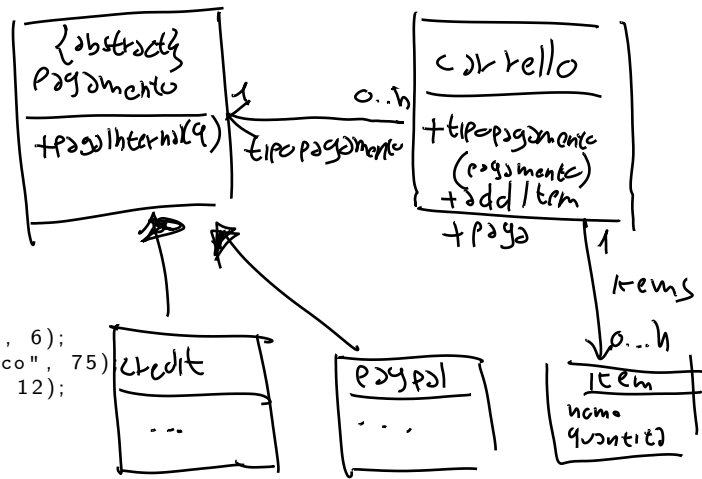
Strategy

8. Si consideri il seguente frammento di codice in Java:

```

1 public class TestAmmazon {
2     public static void main(String[] args) {
3
4         Carrello cart1 = new Carrello();
5
6         Item crema = new Item("Crema da barba", 6);
7         Item rasoio = new Item("Rasoio elettrico", 75);
8         Item dopobarba = new Item("Dopobarba", 12);
9
10        cart1.addItem(crema);
11        cart1.addItem(rasoio);
12        cart1.addItem(dopobarba);
13
14        cart1.removeItem(crema);
15
16        Pagamento pagapaypal = new PayPal("matteo@baldoni.it", "passwd");
17        cart1.tipopagamento(pagapaypal);
18
19        cart1.paga();
20
21        Carrello cart2 = new Carrello();
22
23        Item altracrema = new Item("Crema da barba", 6);
24        Item altrorasoio = new Item("Rasoio a lama", 9);
25        Item altrodopobarba = new Item("Crema lenitiva", 10);
26
27        cart2.addItem(altracrema);
28        cart2.addItem(altrorasoio);
29        cart2.addItem(altrodopobarba);
30
31        Pagamento cc = new CreditCard("Matteo Baldoni", "1234123412341234", "789", "12/21");
32        cart2.tipopagamento(cc);
33
34        cart2.paga();
35
36    }
37 }

```



Questo utilizza un insieme di classi che realizzano un carrello con diverse possibilità per il pagamento per un sito di e-commerce utilizzando un noto pattern GoF. Si dica di quale pattern si tratta e disegnare il diagramma UML delle classi coinvolte.

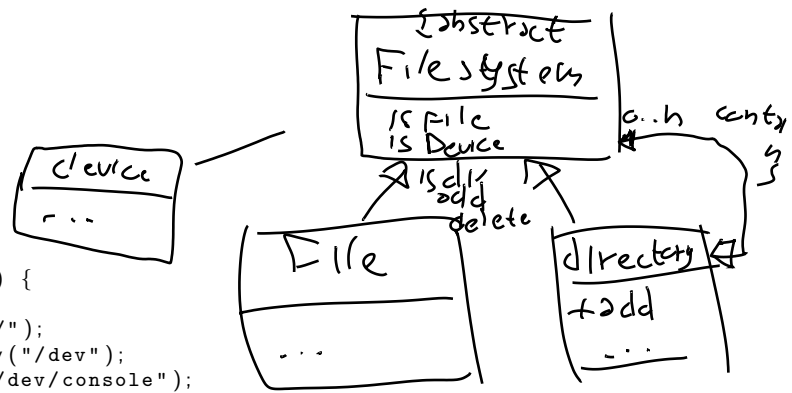


9. Si consideri il seguente frammento di codice in Java:

```

1 public class TestFileSystem {
2     public static void main(String[] args) {
3
4         FileSystem root = new Directory("/");
5         FileSystem devices = new Directory("/dev");
6         FileSystem std_out = new Device("/dev/console");
7         FileSystem home = new Directory("/home");
8         FileSystem baldoni = new Directory("/home/baldoni");
9         FileSystem compitoA = new File("/home/baldoni/compitoA.pdf");
10        FileSystem compitoB = new File("/home/baldoni/compitoB.pdf");
11
12        root.add(devices);
13        root.add(home);
14
15        devices.add(std_out);
16        home.add(baldoni);
17
18        baldoni.add(compitoA);
19        baldoni.add(compitoB);
20        baldoni.delete(compitoA);
21
22        System.out.println("Path: " + compitoB.getName() + ": " + compitoB.getPath());
23
24        System.out.println(std_out.getName() + " is a " + std_out.isFile());
25        System.out.println(std_out.getName() + " is a " + std_out.isDevice());
26        System.out.println(std_out.getName() + " is a " + std_out.isDirectory());
27        System.out.println(compitoB.getName() + " is a " + compitoB.isFile());
28        System.out.println(compitoB.getName() + " is a " + compitoB.isDevice());
29        System.out.println(baldoni.getName() + " is a " + baldoni.isDirectory());
30
31    }
32 }

```



Questo utilizza un insieme di classi che realizzano un file system utilizzando un noto pattern GoF. Si dica di quale pattern si tratta e disegnare il diagramma UML delle classi coinvolte.

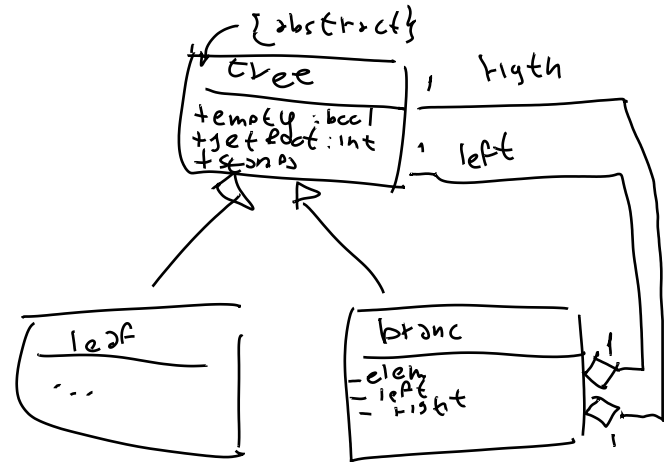


10. Dire a quale pattern GoF la seguente implementaione della struttura dati albero binario si conforma. Disegnare la struttura del pattern e il diagramma delle classi della soluzione applicata alla struttura dati albero binario.

```

1  public abstract class Tree {
2      public abstract boolean empty();
3      public abstract int getRootElem();
4      public abstract void stampapostvisita();
5  }
6
7  public class Leaf extends Tree {
8      public Leaf() { }
9      public boolean empty() {
10         return true;
11     }
12     public int getRootElem() {
13         assert false; return 0;
14     }
15     public void stampapostvisita() { }
16 }
17
18 public class Branch extends Tree {
19     private int elem;
20     private Tree left;
21     private Tree right;
22     public Branch(int elem, Tree left,
23         Tree right) {
24         this.elem = elem;
25         this.left = left;
26         this.right = right;
27     }
28     public boolean empty() {
29         return false;
30     }
31     public int getRootElem() {
32         return elem;
33     }
34     public void stampapostvisita() {
35         right.stampapostvisita();
36         left.stampapostvisita();
37         System.out.print(this.getRootElem()
38             + " ");
39     }
40 }
41
42 public class TestTree {
43     public static void main(String[] args) {
44         ...
45         System.out.println("Stampo albero
46             postvisita");
47         t.stampapostvisita();
48         System.out.println("");
49     }
50 }

```



Risposta:

