PROGRAMMAZIONE II (A,B) - a.a. 2019-20
Prima Verifica Intermedia– 7 Novembre 2019

Domande di base

1. Cosa si intende per *dynamic dispatch* nel contesto dei linguaggi di programmazione a oggetti?
2. Si consideri il seguente programma (scritto con una sitassi Java-like)

```
class Alpha {}

class Beta extends Alpha {}

class Gamma extends Alpha {}

class Epsilon extends Alpha{}

class Fruit extends Gamma{}

class Golf extends Beta {}

class Orange extends Fruit{}
:
\\ main
Fruit f = new Fruit();
Alpha a = f;
Gamma g = f;
a = g;
 }
```

   Definire il tipo statico e quello dinamico per ogni variable presente nel programma.

3. Supponiamo di inserire alla fine del codice del `main` le istruzioni seguenti

```
Beta b = f;
a = b;
```

   Il programma così modificato supera il controllo statico dei tipi? Motivare la risposta.

Esercizio 1

Si consideri un tipo di dato astratto *RecentContacts* introdotto per contenere la lista dei contatti di una chat ordinati sulla base dell'ultima interazione (dalla più alla meno recente). L'interfaccia `RecentContacts` include, tra gli altri, i seguenti metodi

- `public void add(String contact)` *il cui effetto è quello di inserire il contatto (descritto dal parametro* `contact`*) in fondo alla lista dei contatti*
- `public void remove(String contact)` *il cui effetto è quello di rimuovere il contatto dalla lista dei contatti*
- `public void notifyInteraction(String contact)` *il cui effetto è quello di modificare l'ordine della lista a seguito dell'interazione (invio o ricezione di un messaggio) da parte del contatto* `contact`

- `public int size ()` *restituisce il numero dei contatti presenti*
- `public String get(int pos)` *il cui effetto è quello di restituire il contatto nella lista alla posizione pos*
- `public String min ()` *restituisce il contatto che ha interazioni meno recenti nella lista*

1. Si definisca la specifica dell'interfaccia `RecentContacts` , indicando l'elemento tipico e per ogni metodo le clausole REQUIRES, MODIFY ed EFFECTS, il valore restituito e le eventuali eccezioni lanciate in dipendenza dei parametri attuali.
2. Si assuma di implementare la classe `MyRecentContacts` con le seguenti variabili d'istanza:

   ```
   private int elements;
   private ArrayList<String> contatti;
   ```

   - Si definiscano la funzione di astrazione e l'invariante di rappresentazione. Dire se esistono proprietà rilevanti per la classe `MyRecentContacts` che non si possono esprimere con l'nvariante di rappresentazione proposto.
   - Si implementi il costruttore ed il metodo `notifyInteraction` verificando che preservino l'invariante di rappresentazione.
3. Si consideri il tipo `Contacts` che mantiene la lista dei contatti della chat senza un ordine preciso (le post-condizioni dei modificatori non specificano un ordine preciso per gli elementi). Il tipo `Contacts` può essere sottotipo di `RecentContacts` secondo il principio di sostituzione? È possibile il contrario? Motivare le proprie risposte.

Esercizio 2

Si consideri la seguente specifica della classe `Cell`

```
class Cell {
  private  int n;

// @requires true
// @modifies this
// @effect post(this.n)  = p
public void set(int p) { ... }

// @requires true
// @result = pre(this.n))
// @effetct pre(this.n) = post(this.n)
  public int get() { ... }

// @equires true
// @modify this
/// @effect post(this.n) > pre(this.n)
  public void inc() { ... }
}
```

- Dire se l'implementazione del metodo `set` definita dal codice `if(p > 0) n = p;` rispetta la specifica. Motivare la risposta.
- Modificare la specifica in modo tale che il valore della variabile di istanza della classe `Cell` sia sempre positivo.