Linguaggi di Programmazione II: M. Faella **Anno Acc. 2021-2022**

Esame del 2 maggio 2022

Tempo: 2h *Università di Napoli "Federico II"*

Nome e Cognome: Matricola:

1	2	3	4	tot
/40	/20	/25	/15	/100

1. *40 punti* [CrowdGrader]
   Implementare la classe parametrica WeightedSet (insieme pesato), che rappresenta un insieme in cui ad ogni oggetto è associato un peso intero.

   Il metodo add aggiunge un oggetto con un dato peso. Ad ogni oggetto può essere associato un unico peso, mentre oggetti diversi possono avere lo stesso peso.

   Il metodo atLeast accetta un peso p e restituisce una *vista* sull'insieme degli oggetti di peso maggiore o uguale di p. Questa vista supporta l'inserimento di nuovi oggetti con add, ma solo se il loro peso è almeno p, altrimenti add deve lanciare un'eccezione.

   Il metodo toString di un WeightedSet deve elencare gli oggetti contenuti, senza il loro peso, ma *in ordine di peso non decrescente*.

   L'implementazione deve rispettare il seguente esempio d'uso.

   Esempio d'uso:

```
WeightedSet<Object> set = new WeightedSet<>();
set.add(Double.valueOf(3.14), 100);
set.add(new Object(), 5);
set.add("Skylar", 50);
set.add("Jesse", 5);
System.out.println(set);
WeightedSet<Object> set10 = set.atLeast(10);
System.out.println(set10);
set.add("Walter", 60);
System.out.println(set);
System.out.println(set10);
```

   Output:

```
[Jesse, java.lang.Object@6b95977, Skylar, 3.14]
[Skylar, 3.14]
[Jesse, java.lang.Object@6b95977, Skylar, Walter, 3.14]
[Skylar, Walter, 3.14]
```

   ---

2. *20 punti*
   Considerare l'interfaccia Predicate<T>:

```
interface Predicate<T> {
  boolean test(T t);
}
```

   Dire quali delle seguenti sono specifiche valide per un comparatore c tra oggetti di tipo Predicate<T>. In caso negativo, dire quali proprietà sono violate e descrivere un controesempio. c.compare(x,y) restituisce (nei casi non elencati, restituisce zero):

   (a) -1 se x.test(...) restituisce sempre falso e y.test(...) restituisce sempre vero;
   1 se y.test(...) restituisce sempre falso e x.test(...) restituisce sempre vero.

(b) -1 se per tutti gli oggetti `t` il valore di `x.test(t)` è l'opposto di `y.test(t)`;
1 se per tutti gli oggetti `t` il valore di `x.test(t)` è uguale a `y.test(t)`.

(c) -1 se l'insieme degli oggetti `t` per cui `x.test(t)` restituisce vero è un sottoinsieme proprio dell'insieme degli oggetti per cui `y.test(t)` restituisce vero;
1 se esiste un oggetto `t` tale che `x.test(t)` restituisce vero e `y.test(t)` restituisce falso.

(d) -1 se ci sono almeno 10 oggetti diversi su cui `x.test(t)` restituisce vero e `y.test(t)` restituisce falso;
1 se ci sono almeno 10 oggetti diversi su cui `x.test(t)` restituisce falso e `y.test(t)` restituisce vero.

3. *25 punti*
Date le seguenti classi:[1]

```
public class A {
  private A other;
  public A(A other) {
    this.other = other;
  }
  public class B {
    private static int counter = 0;
    private int id = counter++;
  }
  public Object makeObj(int val) {
    return new B() {
      private int j = val;
   };
  }
}
```

Disegnare il *memory layout* che risulta al termine dell'esecuzione del seguente frammento di codice, evidenziando gli eventuali riferimenti impliciti, le variabili catturate e i loro valori:

```
A a1 = new A(null);
A a2 = new A(a1);
A.B b = a1.new B();
Object x = a1.makeObj(42);
A.B y = (A.B) a2.makeObj(42);
```

4. *15 punti*
Dire quali delle seguenti affermazioni sono vere, e quali false. Valutazione: risposta giusta +3 punti, risposta errata -3 punti. Se il totale è negativo, l'esercizio vale 0.

Vero Falso

☐ ☐ `ArrayList<Integer>` è sottotipo di `List<? extends Number>`

☐ ☐ `Set<? extends Number>` è sottotipo di `Set<? super Number>`

☐ ☐ `Map<String,? extends Number>` è sottotipo di `Map<Object,?>`

☐ ☐ `TreeSet<Integer>` è sottotipo di `SortedSet<? super Integer>`

☐ ☐ `HashMap<Integer,Double>` è sottotipo di `Map<?,? super Double>`

[1] A partire da Java 16, le classi interne possono avere attributi statici non costanti.