Esercizio 1.
Si progetti una classe `InsiemeLimitato` che rappresenti un insieme di al più `n` elementi di tipo intero. La classe `InsiemeLimitato` utilizza un array di interi di lunghezza `n` per memorizzare gli elementi dell'insieme ed una variabile `numElementi` per tenere traccia del numero di elementi contenuti nell'insieme.

Si progetti un costruttore con un parametro `n` che indica il numero massimo di elementi che l'insieme può contenere.

Si progettino i seguenti metodi della classe `InsiemeLimitato`:
- `getNumElementi()`: restituisce il numero di elementi contenuti nell'insieme;
- `getSize()`: restituisce il numero massimo di elementi che l'insieme può contenere;
- `boolean contenuto(int e)`: restituisce `true` se l'elemento `e` è contenuto nell'insieme, `false` altrimenti;
- `boolean vuoto()`: restituisce `true` se l'insieme è vuoto, `false` altrimenti;
- `boolean pieno()`: restituisce `true` se l'insieme è pieno, `false` altrimenti;

Si progetti una classe di test che contenga i seguenti metodi statici:

- `boolean confronta(InsiemeLimitato i, InsiemeLimitato j)`: restituisce `true` se gli insiemi `i` e `j` hanno gli stessi elementi, `false` altrimenti;

- `InsiemeLimitato differenza(InsiemeLimitato i, InsiemeLimitato j)`: restituisce l'insieme differenza formato da tutti gli elementi che appartengono ad `i` e non appartengono a `j`.

Infine si scriva il metodo main nel quale si creano due insiemi e si chiamano tutti i metodi.

Esercizio 2.
Si progettino una eccezione `EccezioneInsiemeLimitato` e due metodi `inserisci` e `max` da aggiungere alla classe `InsiemeLimitato`:

- `void inserisci(int e)`: inserisce l'elemento `e` nell'insieme. Se l'insieme è pieno solleva l'eccezione `EccezioneInsiemeLimitato`. (Nota: ovviamente se un elemento appare già nell'insieme non deve essere inserito nuovamente).

- `int max()`: restituisce il massimo dell'insieme. Se l'insieme è vuoto solleva l'ecezione `EccezioneInsiemeLimitato`.

Esercizio 3.
Si dica cosa stampa il seguente programma, motivando la risposta.

```
public class A {

	private int n;
```

```
	public A(int n) {
		this.n=n;
	}

	public void raddoppia() {
		n=n*2;
	}

	public int getN() {
		return n;
	}
}

public class B extends A{

	public B(int n) {
		super(n);
	}

	public void raddoppia() {
		super.raddoppia();
		super.raddoppia();
	}
}

public class Test {

	public static void main(String[] args) {
		A[] vettore = new A[3];
		vettore[0] = new A(1);
		vettore[1] = new B(1);
		vettore[2] = vettore[0];

		for(int i=0;i<vettore.length;i++) {
			A a = vettore[i];
			a.raddoppia();
			System.out.println(a.getN());
		}
	}
}
```