```
CHIARANI FABIO
A.A. 2017-2018

// === Classes ===
  public class Class_name {
    public class_name() {}
  };

  public abstract class Class_name {
    public class_name() {}
    public abstract method_name();
  };

  public interface Interface_name {
    // no constructor
    public void method_name();
  };

  public class Class_name extends Super_class_name {
    public class_name {}
  };

  public abstract class Class_name extends Super_class_name {
    public class_name {}
  };

  public class Class_name implements Interface_name {
    public class_name {}
  };

  public abstract class Class_name implements Interface_name {
    public class_name {}
  };
```

// === Types ===

```
// STATIC: posso istanziarlo più volte
static int value = 5;

// FINAL: posso istanziarlo 1 volta nel costruttore, ma posso usare
// comunque i suoi metodi
// aka 1 sola volta 'new *'
final int value = 5;
```

// === Main method ===

```
  public static void main(String[] args) {
```

```
    // code here!
  }
```

// === Iterable ===

```
import java.util.Iterator;

class Items implements Iterable<Item> {
  private Item[] items;

  @Override
  public Iterator<Item> iterator() {
    return new Iterator<Item> {
      private int pos;

      public boolean hasNext() {
        return pos < items.length; // items is a private array
      }

      public Item next() {
      return items[pos];
      }
    }
  }
}
```

// === Comparable ===
// when comparing, use this instance's fields as first operand
// this > other -> >0
// this == other -> =0
// this < other -> <0

```
class Date implements Comparable<T> {

  @Override
  public int compareTo(T other)
  {
    int diff = this.name.compareTo(other.name);
    if(diff != 0)
    {
      return diff;
    }
    return this.version - other.version;
  }
}
```

// === I.E. Constructor ===

```
protected method_name(String... words)
{
  String[] tmp = words;
}

protected method_name(Iterable<String> words)
{
  ArrayList<String> tmp = new ArrayList<String>();
  for (String w: words)
            list.add(w);

    return list.toArray(new String[list.size()]);
}
```

/* === P - P - p - I ===

```
| Class | Package | Subclass | Subclass | World
|       |         |(same pkg)|(diff pkg)|
————————————+———————+——————————+——————————+———————————+———————
public      |   +   |    +     |    +     |     +     |   +
————————————+———————+——————————+——————————+———————————+———————
protected   |   +   |    +     |    +     |     +     |
————————————+———————+——————————+——————————+———————————+———————
no modifier |   +   |    +     |    +     |           |
————————————+———————+——————————+——————————+———————————+———————
private     |   +   |          |          |           |

+ : accessible
blank : not accessible
```

*/

// === Eccezioni ===

```
public class EmptyWordException extends IllegalArgumentException {
    public EmptyWordException() {
        super("An empty word has been provided");
    }
}

public abstract class QuoteException extends BookingException {
    protected QuoteException(String message) {
        super(message);
    }
}
```

// === Random ===

```
// Proprietà di classe sempre private e final se possibile

// Foreach
for(type name : iterator)
  // some stuff


import java.util.Scanner;
Scanner scanner = new Scanner(System.in);
int n = scanner.nextInt();
String s = scanner.next(); // just one word
String s = scanner.nextLine(); // entire line
boolean b = scanner.nextBoolean();


// === Random ===
import java.util.Random;
Random random = new Random();
random.nextInt(4); // 0 - 3
random.nextDouble();
random.nextBoolean();

// Ricerca numero casuale tra un massimo e un minimo (entrambi
inclusi)
int randomNum = random.nextInt((max - min) + 1) + min;

// === Metodi statici classe Character ===
bool:Character.isDigit(char);
bool:Character.isAlphabetic(char);
bool:Character.isLetter(char);
bool:Character.isLetterOrDigit(char);
bool:Character.isLowerCase(char);
bool:Character.isUpperCase(char);
bool:Character.isWhitespace(char);
char:Character.toLowerCase();
char:Character.toUpperCase();

// === Metodi stringhe ===
String s = "";
char:s.chartAt(index);
String:s.concat(anotherStr);
bool:s.contains(anotherStr);
int:s.indexOf(char);
bool:s.isEmpty();
int:s.length();
Strung:s.replace(oldChar, newChar);
String[]:s.split(str);
String:s.toLowerCase();
String:s.toUpperCase();
String:s.trim(); //toglie gli spazi all'inizio e alla fine
```

```
// === Esempi formattazione stringhe ===
String.format("%c, %d, %f", char, int, float);
String.format("|%20d|", 93); // -> |                   93|
String.format("|%-20d|", 93); // -> |93                   |
String.format("|%020d|", 93); // -> |00000000000000000093|
String.format(".2f", 0.3333); // -> 0.33

// === Metodi generici applicabili a Set e SortedSet ===
bool:set.add(object);
bool:set.addAll(anotherCollection);
void:set.clear();
bool:set.contains(object);
bool:set.containsAll(anotherCollection);
bool:set.equals(object);
int:set.hashCode();
bool:set.isEmpty();
Iterator<T>:set.iterator();
bool_set.remove(object);
bool:set.removeAll(anotherCollection);
bool:set.retainAll(anotherCollection); // Cancella tutto eccetto gli
// elementi contenuti in "anotherCollection"
int:set.size();
T[]:set.toArray(T[] array);


// Metodi applicabili a List
<T>:list.get(index);
bool:list.add(object);
bool:list.add(index, object);
int:list.indexOf(object);
int:list.lastIndexOf(object); // Ritorna l'indice dell'ultimo elemento
//della lista uguale all'oggetto passato come parametro
bool:list.remove(index); // left-shift


// Metodi applicabili al Map
value(che ho aggiunto):map.put(key, value);
void:map.putAll(anotherMap);
value(added)map.putIfAbsent(key, value);
value:map.get(key);
value(rimossa):map.remove(key);
bool:map.containsKey(key);
bool:map.containsValue(value);
<Set>T:map.keySet(); // Ritorna una collezione di chiavi
<Set>T:map.values(); // Ritorna una collezione di valori

// Metodi statici applicabili ad array
Arrays.asList(a, b, c); // crea una lista ("ciao","come")
Arrays.toString(array);

// Lettura da file
// NOTA: ricordarsi di gestire (o rimandare la gestione) di
FileNotFoundException e IOException
import java.io;
InputStream is = new BufferedInputStream(new
```

```
FileInputStream(path));

OutputStream os = new BufferedOutputStream(new
FileOutputStream(path));

int nextByte;
while ((nextByte = is.read()) != -1) {
  os.write(nextByte);
}
is.close();
os.close();


public int getQuoteFor(Date when) throws QuoteException {
        if (when.compareTo(start) < 0 || when.compareTo(end) > 0)
            throw new IllegalBookingDatesException();

        Aircraft aircraft = fleet.getAircraftFor(when);
        int alreadyBooked = getSeatsBooked(when);
        int capacity = aircraft.getCapacity();
        if (alreadyBooked == capacity)
            throw new FlightSoldOutException();

        return minimalPrice + alreadyBooked * (maximalPrice - minimalPrice) /
     (capacity - 1);
}
```