Istruzioni

- Creare una cartella sul desktop con il proprio cognome.
- Aprire eclipse. Scegliere come workspace la cartella creata sul Desktop. Se eclipse si apre senza far scegliere il workspace, una volta avviato impostare File/Switch Workspace/Other... e quindi poter facilmente scegliere personalmente selezionare il workspace.

  ATTENZIONE: se il workspace non è sul desktop tutto il lavoro sarà perso e non recuperabile.
- Creare un **progetto Java** e denominarlo con il proprio **numero di matricola**.
- Al termine del compito, **non** chiudere eclipse e chiamare il docente per consegnare.
- Consegnare solo la cartella del progetto Java. Dentro la cartella creata sul desktop (quella con il cognome) c'è un'altra cartella (quella con il numero di matricola) che corrisponde al progetto Java: consegnare quest'ultima cartella. Chiudere eclipse solo dopo aver consegnato.

1 Tombola [pt. 2]

Scrivere una classe Tombola per modellare un gioco della tombola. I numeri che possono essere estratti sono 90: gli interi da 1 a 90. Ogni giocatore ha una cartella (`cartella[][]`) con 3 righe e 5 colonne contenete 15 numeri distinti.

L'estrazione corrisponde a un numero, non ancora estratto, che va da 1 a 90. I numeri estratti sono modellati con un array booleano (`estratti[]`) di dimensione 90. La cella in posizione i dell'array `estratti[]` è true se e solo se il numero $i + 1$ è stato estratto.

*Il costruttore della classe (`Tombola(int numeri[][])`) deve creare e inizializzare la matrice `cartella[][]` **copiandoci** i numeri passati come parametro (dalla matrice 3x5 di interi `numeri[][]`). Inoltre deve creare e inizializzare l'array (`estratti[]`) a `false` .*

Metodo estrazione La classe dispone di un metodo booleano `estrazione(int numero)` che estrae il `numero` passato per parametro; il metodo è mostrato in Codice 1.

```
public boolean estrazione(int numero) {
    if(numero <=90 && numero >= 1) {
        estratti[numero-1] = true;
        return true;
    }
    return false;
}
```

Codice 1: Metodo `estrazione(int numero)`

Il metodo esegue aggiorna `estratti[]` solo se `numero` è un numero corretto da 1 a 90: in tal caso il *metodo ritorna* `true`, altrimenti ritorna `false`.

Metodo ambo La classe deve disporre di un metodo booleano `ambo()` che controlla se la cartella contiene *un ambo. L'ambo* corrisponde a due numeri estratti presenti **sulla stessa riga** della cartella. Se c'è *almeno una riga* che contiene due numeri estratti allora il metodo restituisce `true`, altrimenti restituisce ***`false`***.

2 JML

Scrivere in JML la seguente postcondizione al costruttore:

1. *ogni cella della cartella è uguale a quella corrispondente della matrice* (`numeri[][]`) passata per *parametro.* **[pt. 0.50]**

Scrivere in JML la seguente precondizione al metodo *estrazione*:

2. *il* `numero` da estrarre, se è un numero corretto da 1 a 90, non è già stato estratto in precedenza. *Utilizzare l'array* `estratti[]`.**[pt. 0.75]**

Scrivere in JML le seguenti postcondizioni al metodo *estrazione*:

3. *dopo l'esecuzione del* metodo *estrazione*, se `numero` è un numero corretto da 1 a 90 allora l'array `estratti[]` è aggiornato di conseguenza; **[pt. 0.75]**
4. *il metodo estrazione*, se `numero` non è un numero corretto da 1 a 90, restituisce `false`. **[pt. 0.50]**

Scrivere in JML la seguente postcondizione al metodo *ambo*:

5. *se la cartella contiene* un ambo **sulla prima riga**, il metodo `ambo()` restituisce `true`. **[pt. 1.50]**

***ATTENZIONE**: si ricorda che* OpenJML non compila i quantificatori con più indici: è possibile però *trasformarli in quantificatori annidati.* Ad esempio:

$(\backslash forall\ \text{int}\ i\,j; i >= 0\ \&\&\ i < n\ \&\&\ j >= 0\ \&\&\ j < m; \ldots))$

può essere trasformato in:

$(\backslash forall\ \text{int}\ i; i >= 0\ \&\&\ i < n; (\backslash forall\ \text{int}\ j; j >= 0\ \&\&\ j < m; \ldots))$

Una condizione in cui compare un quantificatori con più indici che non compila con OpenJML, ma che rispetta la richiesta del testo dell'esame, sarà comunque giudicata come corretta ai fini dell'esame.

3 Junit

In JUnit, scrivere i seguenti casi di test nella classe `TombolaTest`.

2 JML

Scrivere in JML la seguente postcondizione al costruttore:

1. ogni cella della cartella è uguale a quella corrispondente nella matrice (numeri[][]) passata per parametro. [pt. 0.50]

Scrivere in JML la seguente precondizione al metodo estrazione:

2. il numero da estrarre, se è un numero corretto da 1 a 90, non è già stato estratto in precedenza. Utilizzare l'array `estratti[]` [pt. 0.75]

Scrivere in JML le seguenti postcondizioni al metodo estrazione:

3. dopo l'esecuzione del metodo *estrazione*, se numero è un numero corretto da 1 a 90 allora l'array `estratti[]` è aggiornato di conseguenza. [pt. 0.75]
4. il metodo *estrazione*, se **numero** non è un numero corretto da 1 a 90, restituisce `false`. [pt. 0.50]

Scrivere in JML la seguente postcondizione al metodo ambo:

5. se la cartella contiene un ambo **sulla prima riga**, il metodo `ambo()` restituisce `true`. [pt. 1.50]

ATTENZIONE: si ricorda che OpenJML non compila i quantificatori con più indici: è possibile però trasformarli in quantificatori annidati. Ad esempio:

(\forall int $i, j; i >= 0$ && $i < n$ && $j >= 0$ && $j < m; \ldots$))

può essere trasformato in:

(\forall int $i; i >= 0$ && $i < n$; (\forall int $j; j >= 0$ && $j < m; \ldots$))

Una condizione in cui compare un quantificatori con più indici che non compila con OpenJML, ma che rispetta la richiesta del testo dell'esame, sarà comunque giudicata come corretta ai fini dell'esame.

3 Junit

In JUnit, scrivere i seguenti casi di test nella classe `TombolaTest`.

1. per il metodo *estrazione*, scrivere un caso di test in cui si mostri che l'estrazione di un numero mai estratto viene eseguita correttamente, basandosi sul valore ritornato dal metodo; **[pt. 0.75]**
2. per il metodo *estrazione*, scrivere un caso di test in cui si mostri che l'estrazione di un numero estratto in precedenza viene eseguita in modo errato, basandosi sul valore ritornato dal metodo. Il caso di test si aspetta che, se si prova a estrarre un numero già estratto, il metodo dovrebbe restituire `false`. **[pt. 0.75]**
3. scrivere un caso di test che evidenzi che all'inizio l'array `estratti[]` è tutto a `false`. Rendere l'array `estratti[]` pubblico per poterlo testare e utilizzare `assertArrayEquals()`. **[pt. 0.75]**
4. scrivere un caso di test che evidenzi che è possibile costruire una cartella e fare delle estrazioni in modo da ottenere un ambo. Utilizzare il metodo ambo() per verificare l'ambo. **[pt. 0.75]**

4 Copertura

Scrivere in JUnit una test suite per il metodo `estrazione` che soddisfi la copertura delle condizioni. Creare una classe `TombolaCoperturaCondizioni` e commentare in modo opportuno i singoli casi di test. **[pt. 1.5]**

Totale punti = 2 + 4 + 3 + 1.5 = 10.5