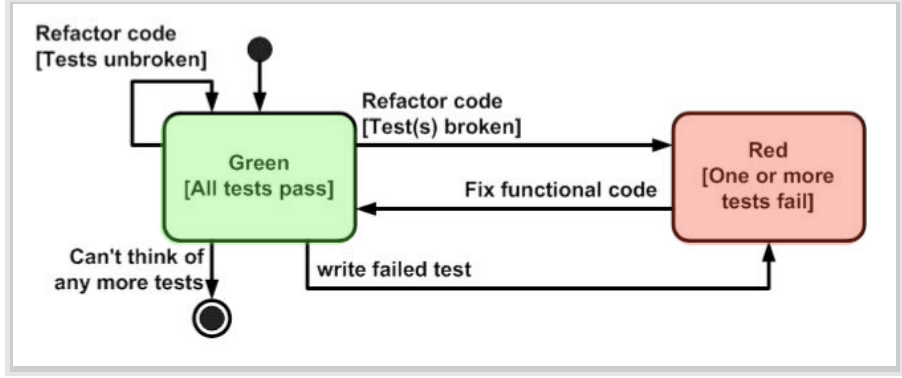
UNIVERSITÀ DEGLI STUDI
DI MILANO

Ingegneria del software

Laboratorio 2: introduzione ai TDD

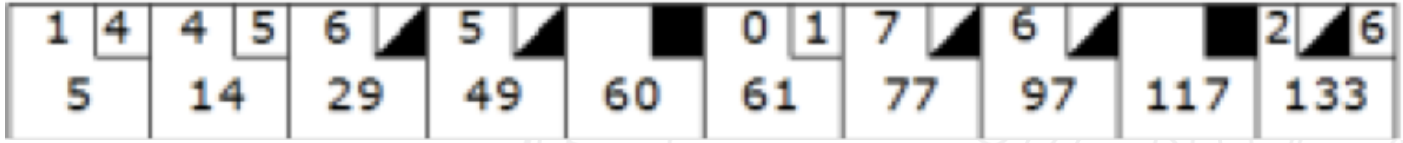
TDD



- Programmate a coppie
- Ogni volta che viene aggiunto un nuovo caso di test, questo deve fallire (rosso) altrimenti se funzionasse non sarebbe la specifica di una funzionalità non sviluppata
- Fate passare il test nella maniera più semplice
  - Se pensate che il caso di test sia un esempio di una situazione più generale e *non siete sicuri* che il vostro codice gestisca un altro caso significativo, aggiungete un test (di triangolazione) che eccezionalmente potrebbe essere a questo punto direttamente verde
- Scambiatevi i ruoli alla tastiera
- Ragionate se ciò che è stato scritto e se ritenete utile migliorate il codice mediante *refactoring*
- fate commit con git ogni volta che aggiungete un test, ogni volta lo fate passare la prima volta, ogni volta che fate refactoring
- Vi diamo noi i casi di test (o almeno una loro descrizione) un po' alla volta

Bowling Kata

- The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.
- A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down) plus a bonus of 5 (the number of pins knocked down on the next roll.)
- A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled.
- In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.



Altre cose da fare

- dare i permessi di *Developer* al prof **Bellettini** e al prof **Monga**
- dare i permessi di *Owner* al **compagno di coppia**
- mettere i **nomi dei due componenti** della coppia nella *descrizione* del progetto (può essere fatto direttamente anche durante l'operazione di fork) e nel file *README.md* sotto al titolo
- implementare quanto richiesto nel *README* e abilitare mano a mano i test
- fare commit delle modifiche e push su gitlab (anche più di una volta)

Lo scopo del laboratorio è cominciare a impratichirsi con scrittura di test e TDD

Fizzbuzz Kata

Esercizio dato a chi ha finito prima.
Createvi da zero un progetto e …

```
public interface IfizzBuzz {
        public String fizzbuzzSequence(int number);
}
```

Implementare una classe `FizzBuzzer` che implementa questa interfaccia e che venga testata da una classe `TestFizzBuzzer`.

Test:

- Se la funzione viene chiamata con valore 1, deve restituire la stringa “1”
- Se viene chiamata con il valore 2, deve restituire la stringa “1 2”
- Se viene chiamata con il valore 3 deve restituire “1 2 fizz”
- Se viene chiamata con il valore 6 deve restituire la stringa “1 2 fizz 4 buzz fizz”
- Se viene chiamata con il valore 15 deve restituire la stringa “1 2 fizz … 13 14 fizzbuzz”

Identifica cioè i multipli di 3 e di 5 e compone la soluzione in maniera appropriata