Tracing & Hypothesis

Bug 3: The DiceValues are the same for each game

1 Tracing

The methods that generate a new DiceValue for each dice occur in the **Main.java**, **Game.java** and **Dice.java**.

Starting from the **Main.java**, three dices of the game are called:

```
BufferedReader console = new BufferedReader(new InputStreamReader(System.in));

Dice d1 = new Dice();
Dice d2 = new Dice();
Dice d3 = new Dice();

Player player = new Player("Fred", 100);
Game game = new Game(d1, d2, d3);
List<DiceValue> cdv = game.getDiceValues();
```

The **Main.java** also implies that the assigning of dice occurs in the *playRound* method of Game.java.

```
while (player.balanceExceedsLimitBy(bet) && player.getBalance() < 200)
{
    turn++;
    DiceValue pick = DiceValue.getRandom();

    System.out.printf("Turn %d: %s bet %d on %s\n",
            turn, player.getName(), bet, pick);

    int winnings = game.playRound(player, pick, bet);
    cdv = game.getDiceValues();

    System.out.printf("Rolled %s, %s, %s\n",
            cdv.get(0), cdv.get(1), cdv.get(2));

    if (winnings > 0) {
        System.out.printf("%s won %d, balance now %d\n\n",
                player.getName(), winnings, player.getBalance());
        winCount++;
```

The rolling of dice occurs in *playRound* method as:

```java
public int playRound(Player player, DiceValue pick, int bet ) {
    if (player == null) throw new IllegalArgumentException("Player cannot be null.");
    if (pick == null) throw new IllegalArgumentException("Pick cannot be negative.");
    if (bet < 0) throw new IllegalArgumentException("Bet cannot be negative.");

    // player.takeBet(bet);

    int matches = 0;
    for ( Dice d : dice) {
        d.roll();
        if (d.getValue().equals(pick)) {
            matches ++;
        }
    }

    int winnings = matches * bet;

    if (matches > 0) {
        player.receiveWinnings(winnings);
    }

    else player.takeBet(bet);
    return winnings;
}
```

The rolling in **Dice.java** occurs as:

```java
public Dice() {
    value =  DiceValue.getRandom();
}

public DiceValue getValue() {
    return value;
}

public DiceValue roll() {
    return DiceValue.getRandom();
}
```

So from this, it could be identified that:

- The *value* variable in **Dice.java** is invariant throughout the game as it is just assigned in the constructor of Dice, but no changing occurs further in the file.
- The compare of the pick is with the *d.getValue()*, which is the value of the Dice originally, rather than the value of the dice after rolling.
- The *roll()* method, although assigns a new DiceValue to the dice, the comparison of DiceValue is with the *getValue()* method, which returns the invariant variable *value.*
- The three dices are set at the beginning of the game in **Main.java**, with the given *value* at the setting of constructor. If the *value* variable in Dice(), the dices may be reused again throughout the game.

2 Hypotheses

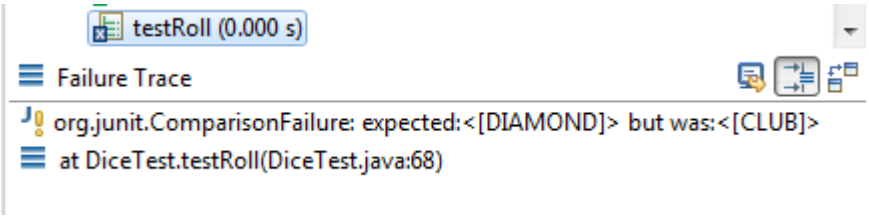
Given the matches are correct (as demonstrated in Bug 1), the hypotheses are:

- **Hypothesis 1**: the *roll()* method does not change the value of DiceValue (incorrect).

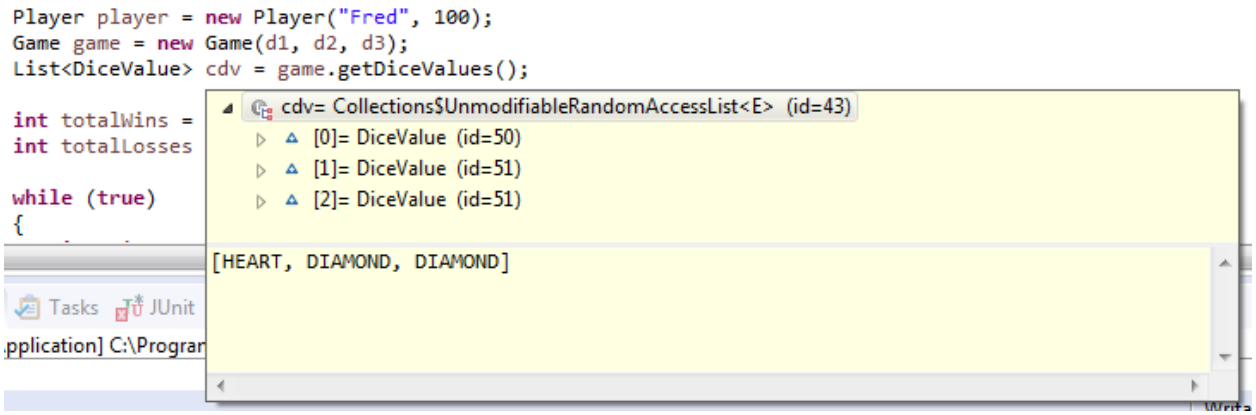
- **Hypothesis 2:** the result of each turn is decided by comparing the pick with the *value* of the dice (correct).
- **Hypothesis 3:** three dices are assigned with each value and reused throughout the game (incorrect).

To test the first hypothesis, a unit test (**DiceTest.java**) is set up to examine the method of getRandom method. Result of the test shows that the *roll()* method is incorrect.

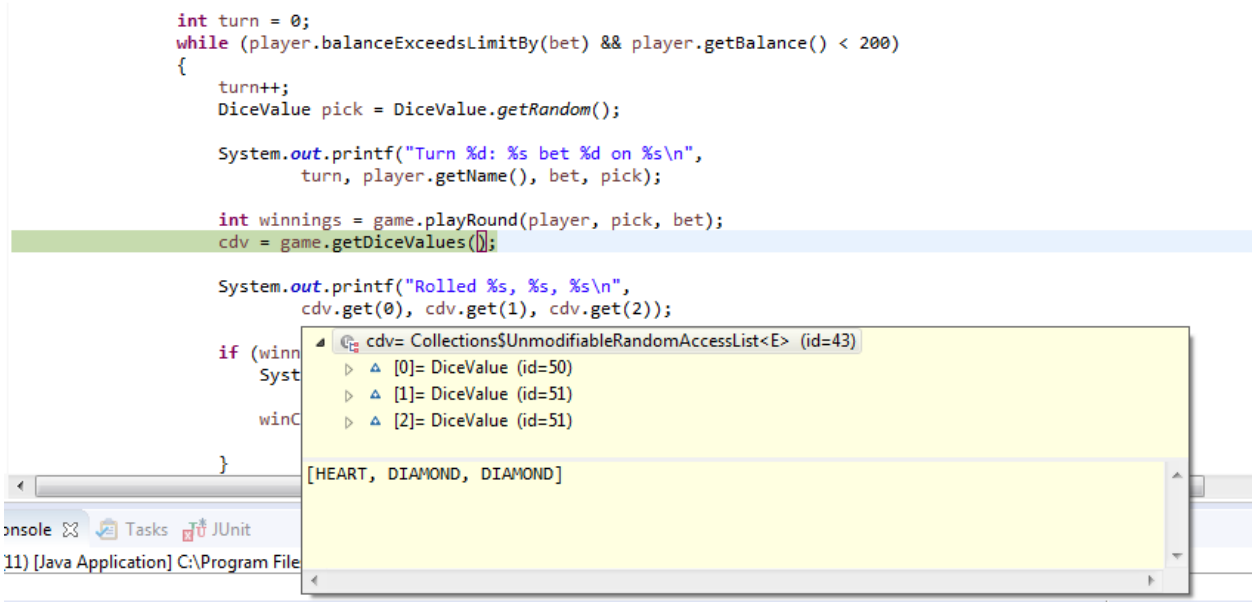
```
@Test
public void testRoll() {
    System.out.println("\nTest Dice roll values ");
    dice = new Dice();
    value = dice.roll().toString();
    System.out.println("Expected: " + value + " | Actual result: " + dice.getValue().toString());
    assertEquals(value, dice.getValue().toString());
}
```



To test the second and third hypotheses, one breakpoint is put within the loop to monitor the changing state of DiceValue. It is verified as at the start of the game, three dice values in *cdv* are:



but after rolling, three dices value are still the same:


int turn = 0;
while (player.balanceExceedsLimitBy(bet) && player.getBalance() < 200)
{
turn++;
DiceValue pick = DiceValue.getRandom();
System.out.printf("Turn %d: %s bet %d on %s\n",
turn, player.getName(), bet, pick);
int winnings = game.playRound(player, pick, bet);
cdv = game.getDiceValues();
System.out.printf("Rolled %s, %s, %s\n",
cdv.get(0), cdv.get(1), cdv.get(2));
if (winn
Syst
winC
}
cdv= Collections$UnmodifiableRandomAccessList<E> (id=43)
[0]= DiceValue (id=50)
[1]= DiceValue (id=51)
[2]= DiceValue (id=51)
[HEART, DIAMOND, DIAMOND]
Tasks
JUnit
11) [Java Application] C:\Program File