Tracing & Hypothesis

Bug 1: The player is not paid out correctly

1 Tracing

From the display of winning in **Main.java**, it can be seen that the winning is calculated in the method *playRound* in **Game.java.** The winning is displayed through the method *getBalance* in **Player.java**. There are probably two possibilities in this case, either problems with the winnings (**Player.java**) or the game (**Game.java**).

```
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++;

}
else {
    System.out.printf("%s lost, balance now %d\n\n",
            player.getName(), player.getBalance());
    loseCount++;
}
```

Tracing from the **Player.java**, it is showed that the balance before the turn and winning after turn are calculated as:

```
public void takeBet(int bet) {
    if (bet < 0) throw new IllegalArgumentException("Bet cannot be zero or negative.");
    if (!balanceExceedsLimitBy(bet)) throw new IllegalArgumentException("Placing bet would go below limit.");
    balance = balance - bet;
}

public void receiveWinnings(int winnings) {
    if (winnings < 0) throw new IllegalArgumentException("Winnings cannot be negative.");
    balance = balance + winnings;
}
```

Tracing from the **Game.java**, the *playRound* method is calculated 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); // refund the bet
    }
    return winnings;
}
```

So the bug here can be first explored that when the player plays round:

- Before the dice rolling, the player has to take the bet: $balance = (initial)\ balance - bet$
- If one match, $balance = balance + bet = (initial)\ balance$ instead of $(initial)\ balance + bet$
- If two matches, $balance = balance + 2 * bet = (initial)\ balance + bet$ instead of $(initial)\ balance + 2 * bet$
- If three matches, $balance = balance + 3 * bet = (initial)\ balance + 2 *\ bet$ instead of $(initial)\ balance + 3 * bet$

2 Hypothesis

There are three hypotheses in this case that needs to be verified:

- **Hypothesis 1:** The number of matches is calculated correctly.
- **Hypothesis 2:** The winnings are added correctly
- **Hypothesis 3:** The balance becomes incorrect after taking the bet

The testing of the hypotheses is conducted by putting breakpoints at

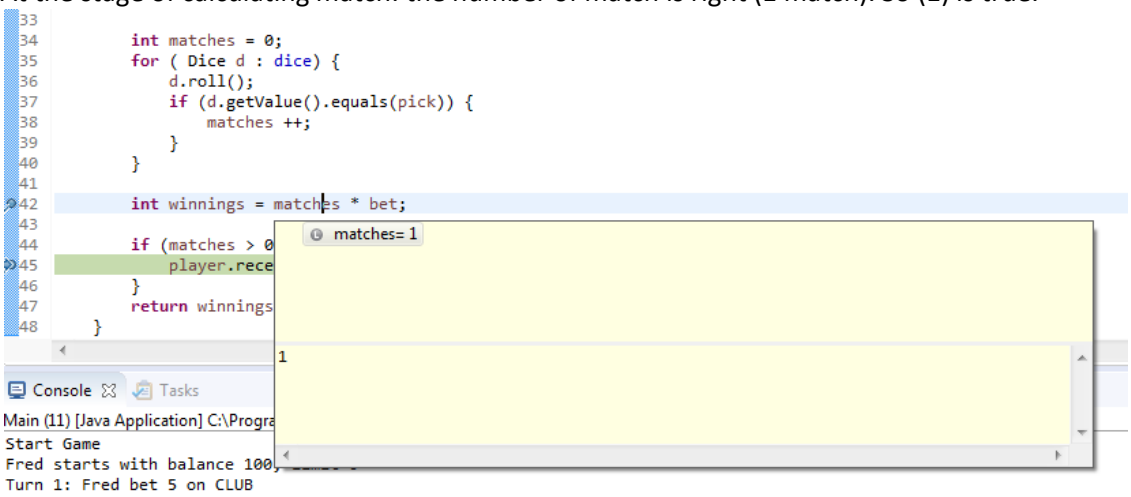
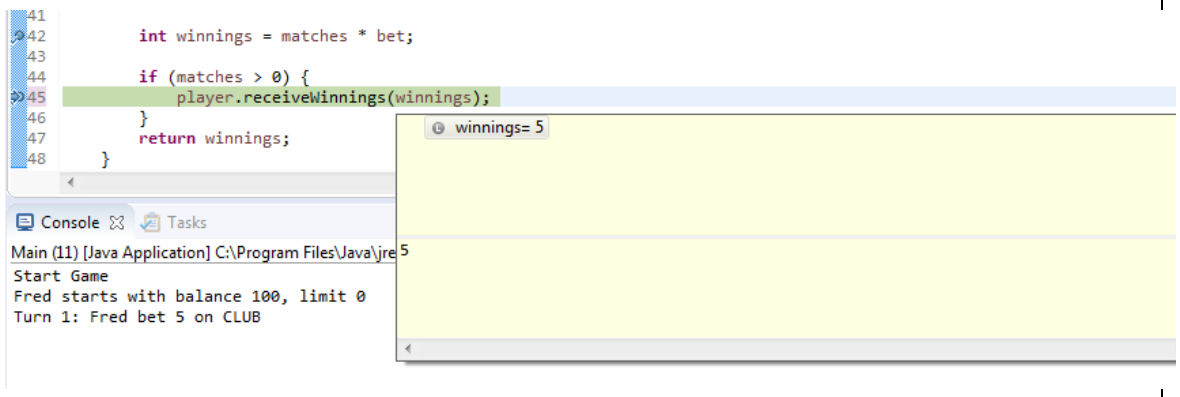
(1) `int winnings = matches * bet;`

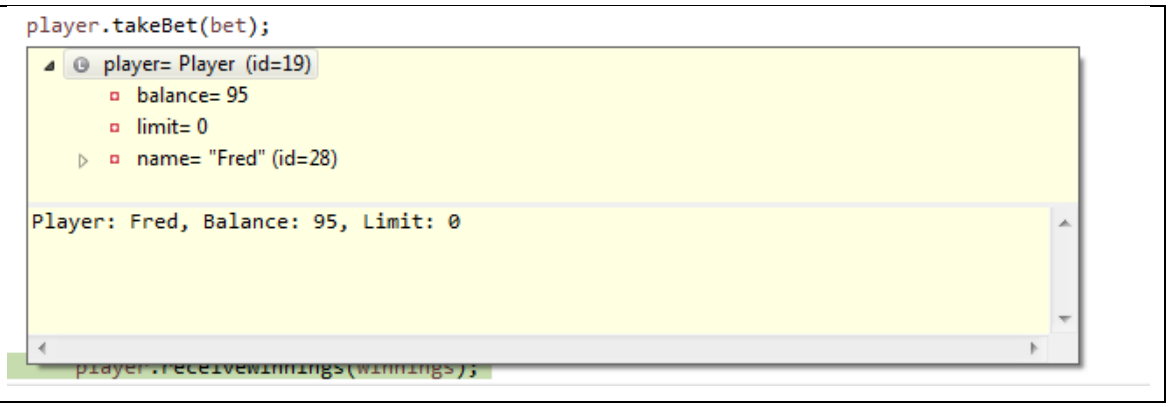
(2) `player.receiveWinnings(winnings);`

The debugging shows the results for the hypotheses as below:

- Result from the console:

```
Start Game
Fred starts with balance 100, limit 0
Turn 1: Fred bet 5 on CLUB
Rolled DIAMOND, CLUB, CROWN
Fred won 5, balance now 100
```

HYPOTHESIS	RESULT
Hypothesis 1: The number of matches is calculated correctly.	At the stage of calculating match: the number of match is right (1 match). So (1) is true. 
Hypothesis 2: The winnings are added correctly	At the stage of calculating winnings. So (2) is true. 
Hypothesis 3: The balance becomes incorrect after taking the bet	Moving to the stage of take bet: the balance of player is wrong. It should be 100. So (3) is true.

	 player.takeBet(bet); player= Player (id=19) balance= 95 limit= 0 name= "Fred" (id=28) Player: Fred, Balance: 95, Limit: 0