Before' and 'after' screen shots identifying the origin of the bug

Bijaya Raj Basnet

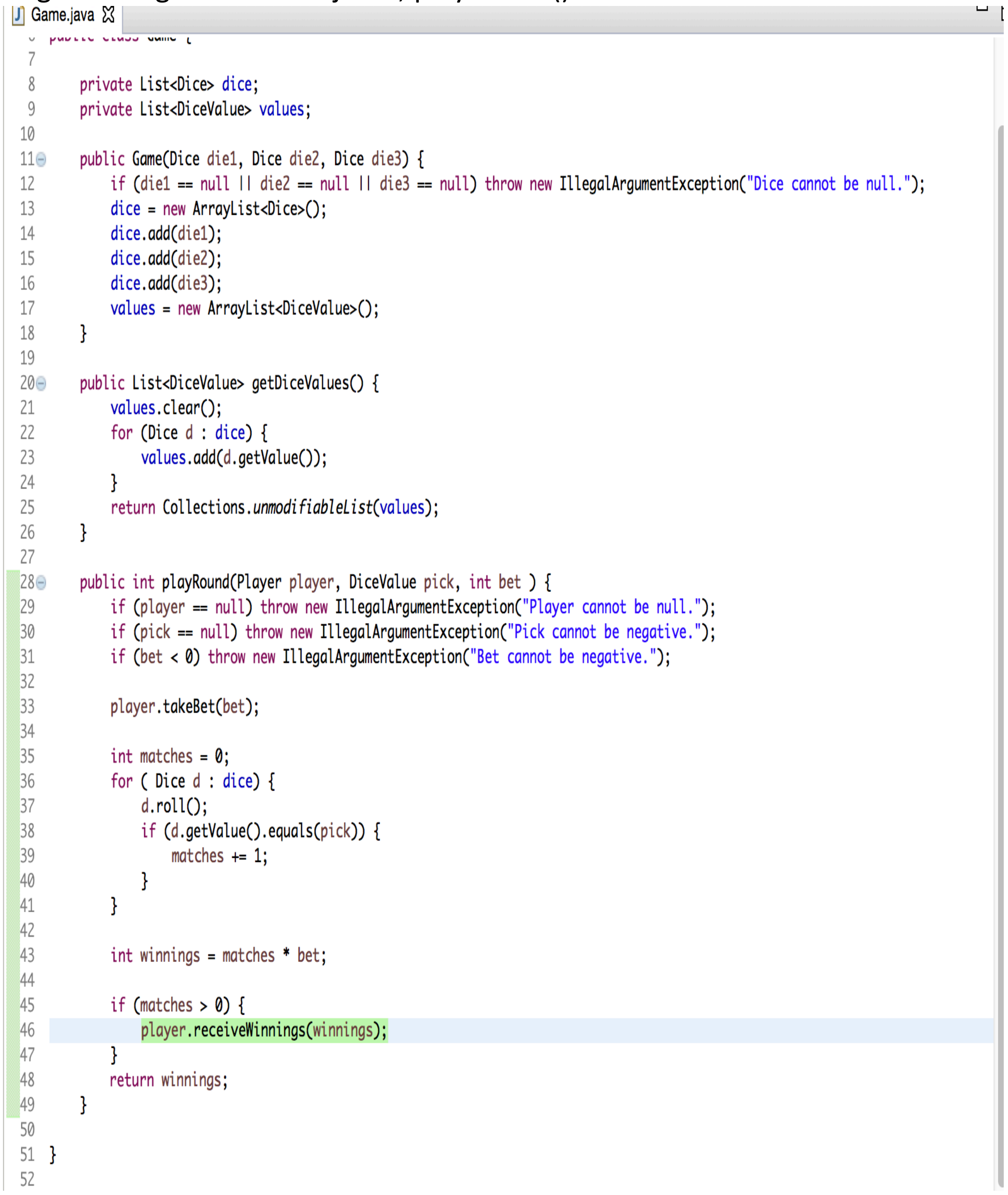
Table of Contents

Before' and 'after' screen shots identifying the origin of the bug
Bijaya Raj Basnet

Bug 1: Incorrect Payout on win

Before Code:

The screenshot shows with the highlighted are shows the origin of incorrect pay bug. The bug is in “Game.java”, playRound() method.



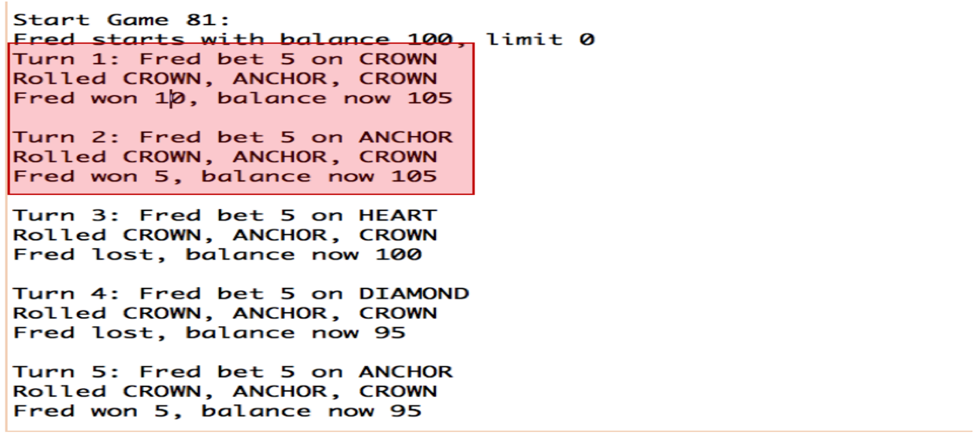
```
Game.java

    private List<Dice> dice;
    private List<DiceValue> values;

    public Game(Dice die1, Dice die2, Dice die3) {
        if (die1 == null || die2 == null || die3 == null) throw new IllegalArgumentException("Dice cannot be null.");
        dice = new ArrayList<Dice>();
        dice.add(die1);
        dice.add(die2);
        dice.add(die3);
        values = new ArrayList<DiceValue>();
    }

    public List<DiceValue> getDiceValues() {
        values.clear();
        for (Dice d : dice) {
            values.add(d.getValue());
        }
        return Collections.unmodifiableList(values);
    }

    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 += 1;
            }
        }

        int winnings = matches * bet;

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

}
```

Fig: Screenshot showing the highlighted code that contains error

Before' and 'after' screen shots identifying the origin of the bug
Bijaya Raj Basnet

Before Output:

The following screenshot shows the incorrect output as the result of the bug in the program. It clearly shows the final balance isn't correct on win.

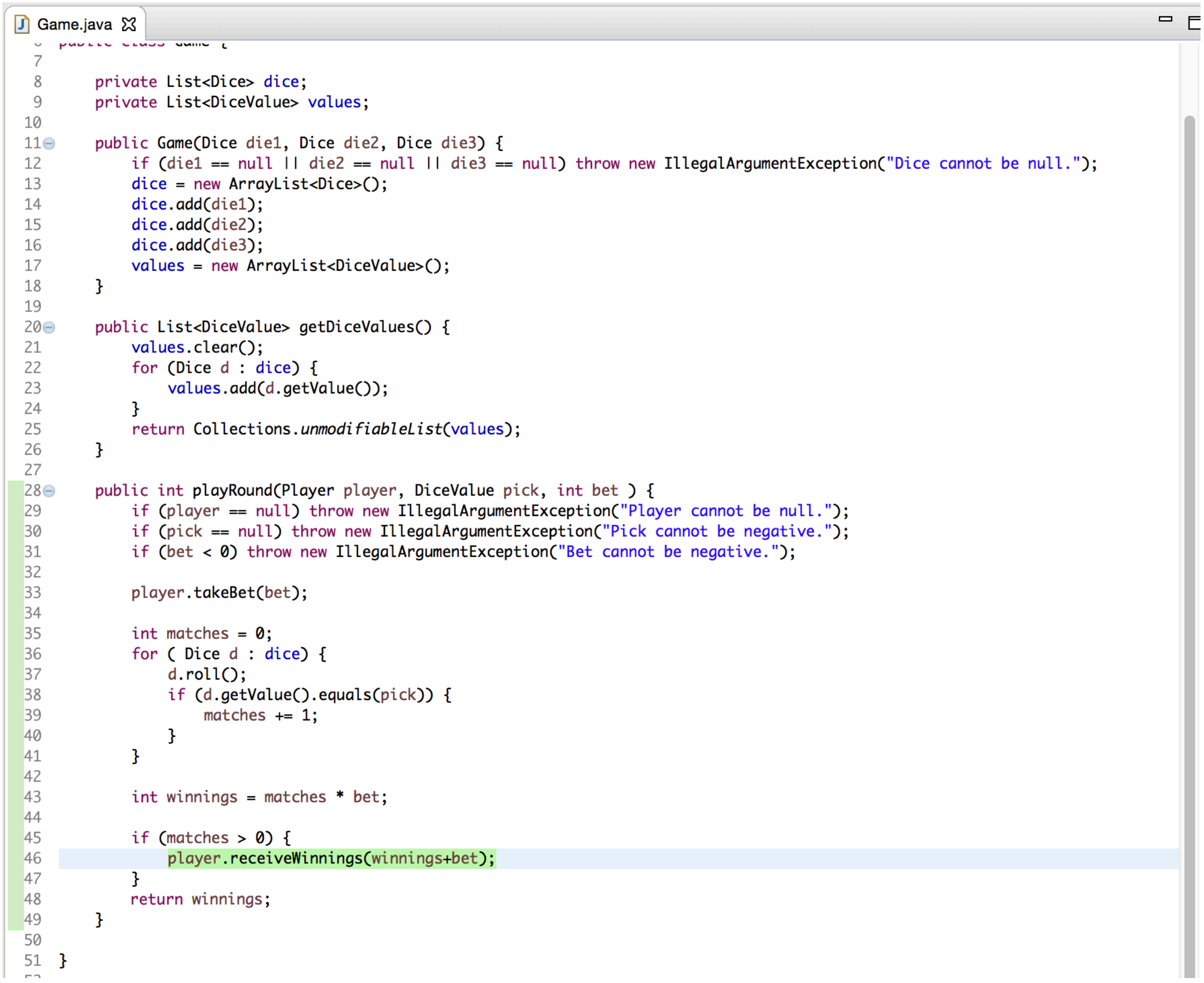


```
Start Game 81:
Fred starts with balance 100, limit 0
Turn 1: Fred bet 5 on CROWN
Rolled CROWN, ANCHOR, CROWN
Fred won 10, balance now 105

Turn 2: Fred bet 5 on ANCHOR
Rolled CROWN, ANCHOR, CROWN
Fred won 5, balance now 105

Turn 3: Fred bet 5 on HEART
Rolled CROWN, ANCHOR, CROWN
Fred lost, balance now 100

Turn 4: Fred bet 5 on DIAMOND
Rolled CROWN, ANCHOR, CROWN
Fred lost, balance now 95

Turn 5: Fred bet 5 on ANCHOR
Rolled CROWN, ANCHOR, CROWN
Fred won 5, balance now 95
```

Fig: screenshot showing incorrect output

After Debugged Code:

The below screenshot shows the highlighted code that has fixed the bug. I have simply returned bet amount back to the player in case of win condition.

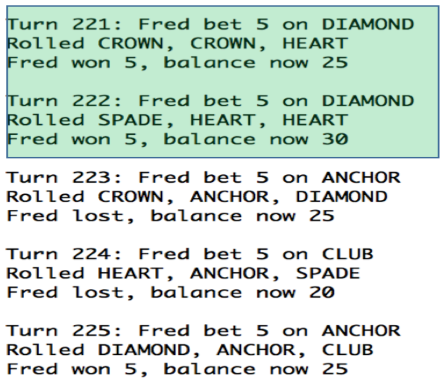


```java

    private List<Dice> dice;
    private List<DiceValue> values;

    public Game(Dice die1, Dice die2, Dice die3) {
        if (die1 == null || die2 == null || die3 == null) throw new IllegalArgumentException("Dice cannot be null.");
        dice = new ArrayList<Dice>();
        dice.add(die1);
        dice.add(die2);
        dice.add(die3);
        values = new ArrayList<DiceValue>();
    }

    public List<DiceValue> getDiceValues() {
        values.clear();
        for (Dice d : dice) {
            values.add(d.getValue());
        }
        return Collections.unmodifiableList(values);
    }

    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 += 1;
            }
        }

        int winnings = matches * bet;

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

}
```

Fig: Screenshot showing the highlighted debugged code

After Debugged Output:

The following screenshot shows the correct output as the result of fixing of the bug in the program. It clearly shows the final balance is correct on win.



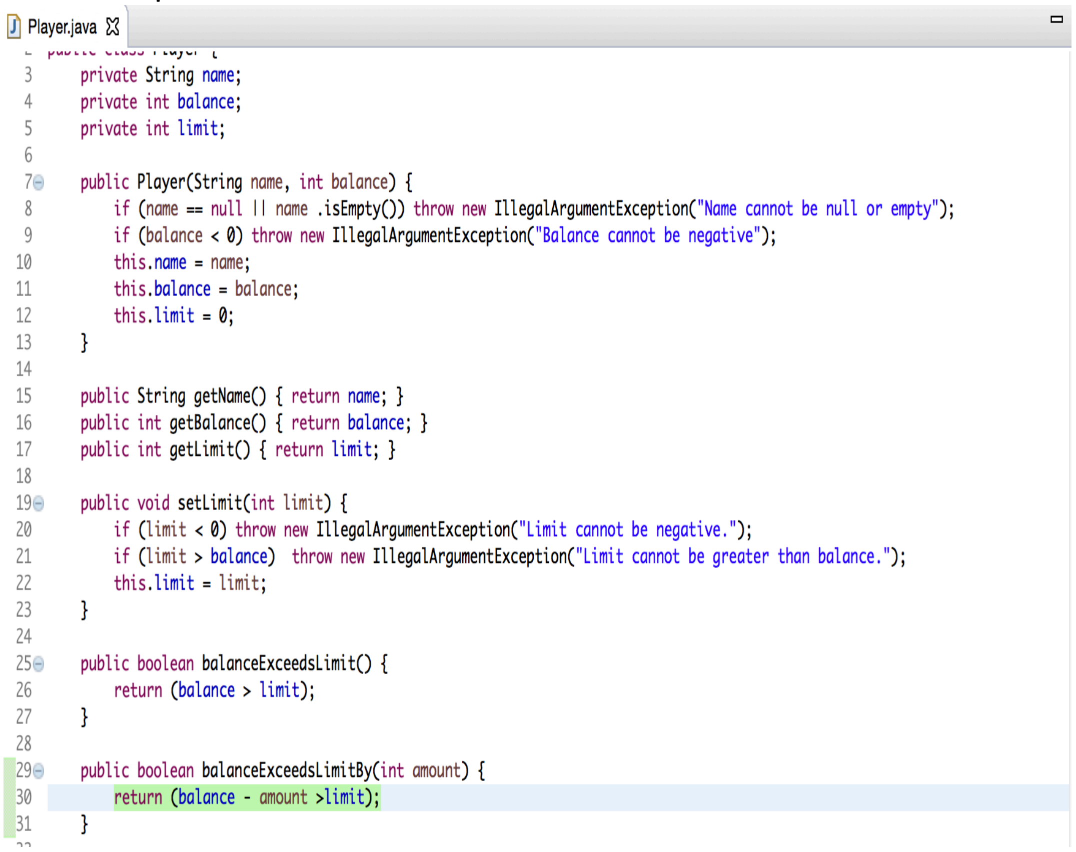
```
Turn 221: Fred bet 5 on DIAMOND
Rolled CROWN, CROWN, HEART
Fred won 5, balance now 25

Turn 222: Fred bet 5 on DIAMOND
Rolled SPADE, HEART, HEART
Fred won 5, balance now 30

Turn 223: Fred bet 5 on ANCHOR
Rolled CROWN, ANCHOR, DIAMOND
Fred lost, balance now 25

Turn 224: Fred bet 5 on CLUB
Rolled HEART, ANCHOR, SPADE
Fred lost, balance now 20

Turn 225: Fred bet 5 on ANCHOR
Rolled DIAMOND, ANCHOR, CLUB
Fred won 5, balance now 25
```

fig: Screenshot showing correct output

BUG 2: Player cannot reach betting limit

Before code:

The following screenshot shows the highlighted code which is the main source of the bug. The bug was present in “Player.java” class, balanceExceedsLimitBy() method. The game only plays if the player balance is greater than bet amount but not equal to bet amount.



Player.java

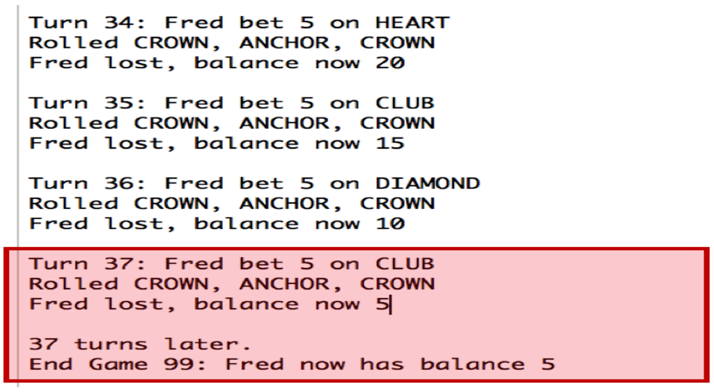
```java
    private String name;
    private int balance;
    private int limit;

    public Player(String name, int balance) {
        if (name == null || name .isEmpty()) throw new IllegalArgumentException("Name cannot be null or empty");
        if (balance < 0) throw new IllegalArgumentException("Balance cannot be negative");
        this.name = name;
        this.balance = balance;
        this.limit = 0;
    }

    public String getName() { return name; }
    public int getBalance() { return balance; }
    public int getLimit() { return limit; }

    public void setLimit(int limit) {
        if (limit < 0) throw new IllegalArgumentException("Limit cannot be negative.");
        if (limit > balance)  throw new IllegalArgumentException("Limit cannot be greater than balance.");
        this.limit = limit;
    }

    public boolean balanceExceedsLimit() {
        return (balance > limit);
    }

    public boolean balanceExceedsLimitBy(int amount) {
        return (balance - amount >limit);
    }
```

Fig: Screenshot highlighting the source of bug.

Before' and 'after' screen shots identifying the origin of the bug
Bijaya Raj Basnet

Before Output:

The following screenshot shows the incorrect output as the result of the bug in the program. It clearly shows the game ends before the balance is 0.

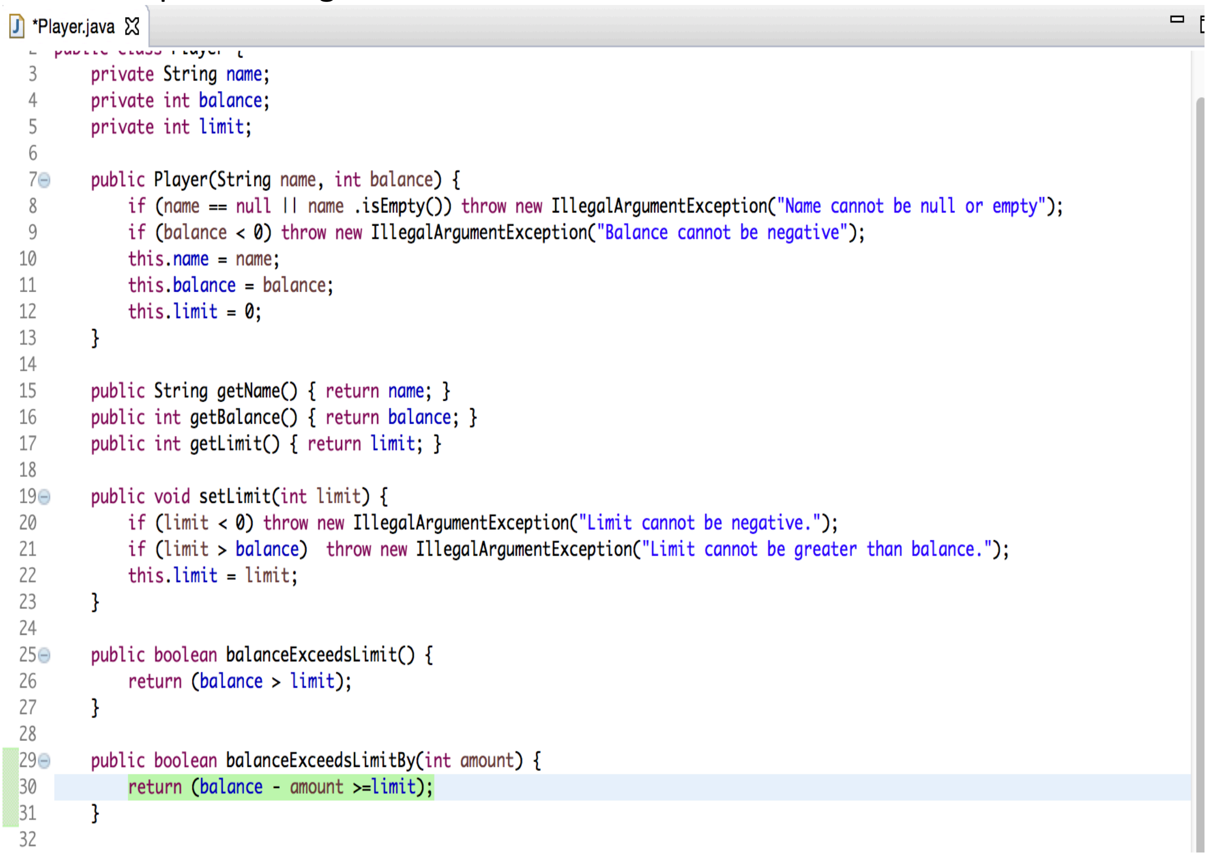


```
Turn 34: Fred bet 5 on HEART
Rolled CROWN, ANCHOR, CROWN
Fred lost, balance now 20

Turn 35: Fred bet 5 on CLUB
Rolled CROWN, ANCHOR, CROWN
Fred lost, balance now 15

Turn 36: Fred bet 5 on DIAMOND
Rolled CROWN, ANCHOR, CROWN
Fred lost, balance now 10

Turn 37: Fred bet 5 on CLUB
Rolled CROWN, ANCHOR, CROWN
Fred lost, balance now 5

37 turns later.
End Game 99: Fred now has balance 5
```

Fig: Screenshot showing incorrect output as a result of bug.

After Debugging code:

The following screenshot shows the highlighted bug free code The bug was present in "Player.java" class, balanceExceedsLimitBy() method. The game only plays if the player balance is greater than bet amount but not equal to bet amount. So I just modified the code so that the game is on when the player balance equals to or greater than bet amount.



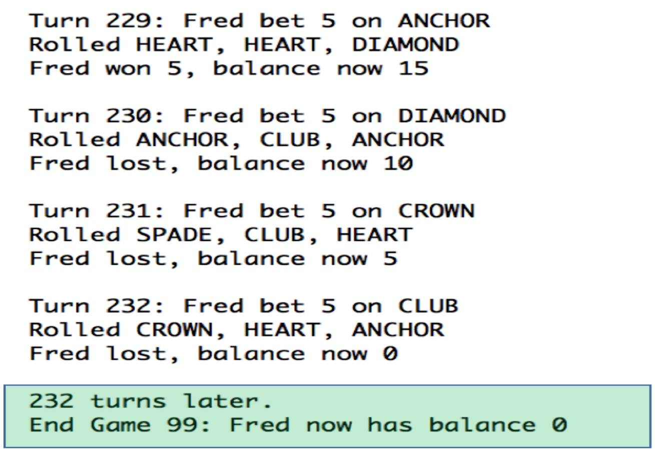
*Player.java

```java
    private String name;
    private int balance;
    private int limit;

    public Player(String name, int balance) {
        if (name == null || name .isEmpty()) throw new IllegalArgumentException("Name cannot be null or empty");
        if (balance < 0) throw new IllegalArgumentException("Balance cannot be negative");
        this.name = name;
        this.balance = balance;
        this.limit = 0;
    }

    public String getName() { return name; }
    public int getBalance() { return balance; }
    public int getLimit() { return limit; }

    public void setLimit(int limit) {
        if (limit < 0) throw new IllegalArgumentException("Limit cannot be negative.");
        if (limit > balance)  throw new IllegalArgumentException("Limit cannot be greater than balance.");
        this.limit = limit;
    }

    public boolean balanceExceedsLimit() {
        return (balance > limit);
    }

    public boolean balanceExceedsLimitBy(int amount) {
        return (balance - amount >=limit);
    }
```

Fig: Screenshot highlighting the fixed code for elimination of bug.

After Debugging output:

The following screenshot shows the correct output as the result of fixing the bug in the program. It clearly shows the game ends when the balance is 0.



```
Turn 229: Fred bet 5 on ANCHOR
Rolled HEART, HEART, DIAMOND
Fred won 5, balance now 15

Turn 230: Fred bet 5 on DIAMOND
Rolled ANCHOR, CLUB, ANCHOR
Fred lost, balance now 10

Turn 231: Fred bet 5 on CROWN
Rolled SPADE, CLUB, HEART
Fred lost, balance now 5

Turn 232: Fred bet 5 on CLUB
Rolled CROWN, HEART, ANCHOR
Fred lost, balance now 0

232 turns later.
End Game 99: Fred now has balance 0
```

Fig: Screenshot showing correct output as a result of fixing the bug.

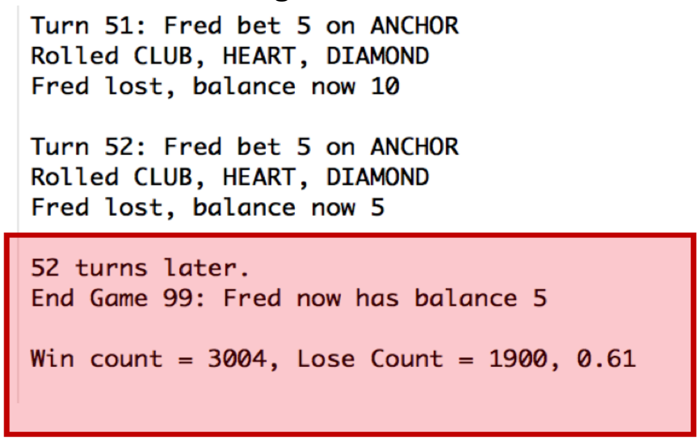
Bug 3: Incorrect Game overall odds

Before Code:

It was found out that this bug is present in the game not because of a single error in the code. It is present because there are two other bugs that results in this bug. The two bugs are looked into bug 4 and bug 5 respectively. Please refer to bug u and bug 5 before code screenshot for further reference.

Before output:

The following screenshot shows the incorrect ratio which is higher than expected (0.42-0.45) as the result of the bug in the program. The ratio here is 0.61 which is too high.



```
Turn 51: Fred bet 5 on ANCHOR
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 10

Turn 52: Fred bet 5 on ANCHOR
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 5

52 turns later.
End Game 99: Fred now has balance 5

Win count = 3004, Lose Count = 1900, 0.61
```

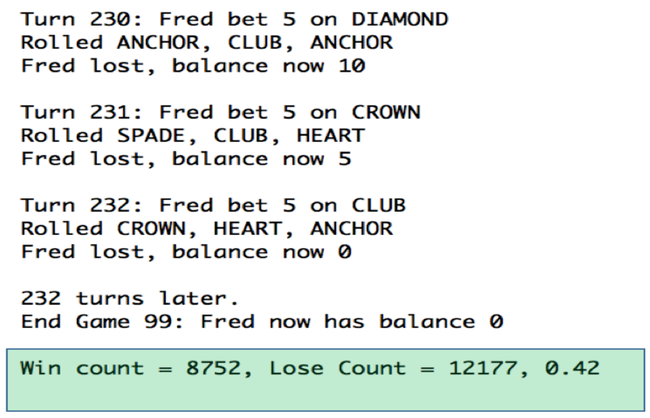
Fig: Screenshot showing incorrect output as a result of bug.

After debugging code:

As described above the source of this bug was because of bug 4 and bug 5 which we will look next. The after debugged code of this bug is also the debugged code of bug4 and bug 5 collectively. Please refer to the after debugged code screenshot of bug 4 and bug 5 for further reference.

After debugging output:

The following screenshot shows the correct output as the result of fixing the bug in the program. The win ratio here is 0.42 which is in line of the expected result.



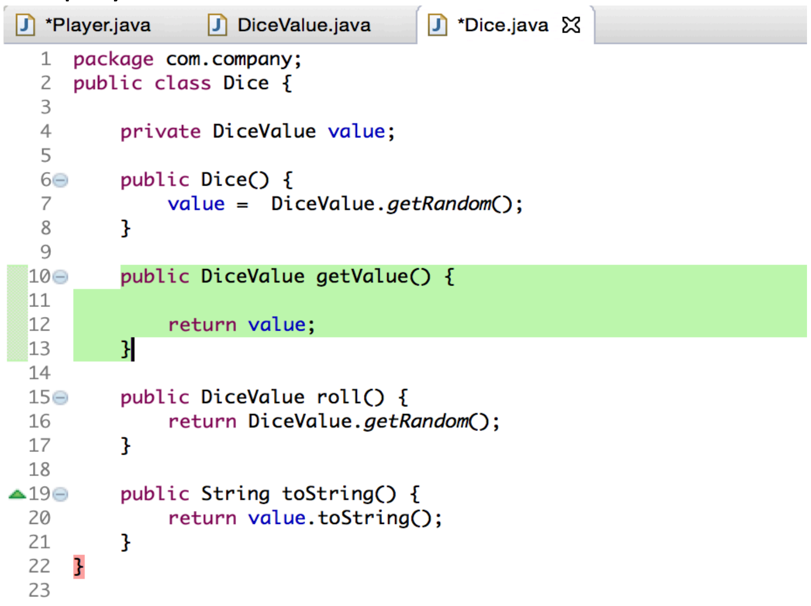
```
Turn 230: Fred bet 5 on DIAMOND
Rolled ANCHOR, CLUB, ANCHOR
Fred lost, balance now 10

Turn 231: Fred bet 5 on CROWN
Rolled SPADE, CLUB, HEART
Fred lost, balance now 5

Turn 232: Fred bet 5 on CLUB
Rolled CROWN, HEART, ANCHOR
Fred lost, balance now 0

232 turns later.
End Game 99: Fred now has balance 0

Win count = 8752, Lose Count = 12177, 0.42
```

Fig: Screenshot showing correct output as a result of fixing the bug.

Bug 4: Dice seem to always roll the same after first roll.

Before Code:

The following screenshot shows the reason of the bug present in the program. The bug is present in Dice.java class, DiceValue getValue() method. It is present there because the method is returing the same values of dice on every round of the play.

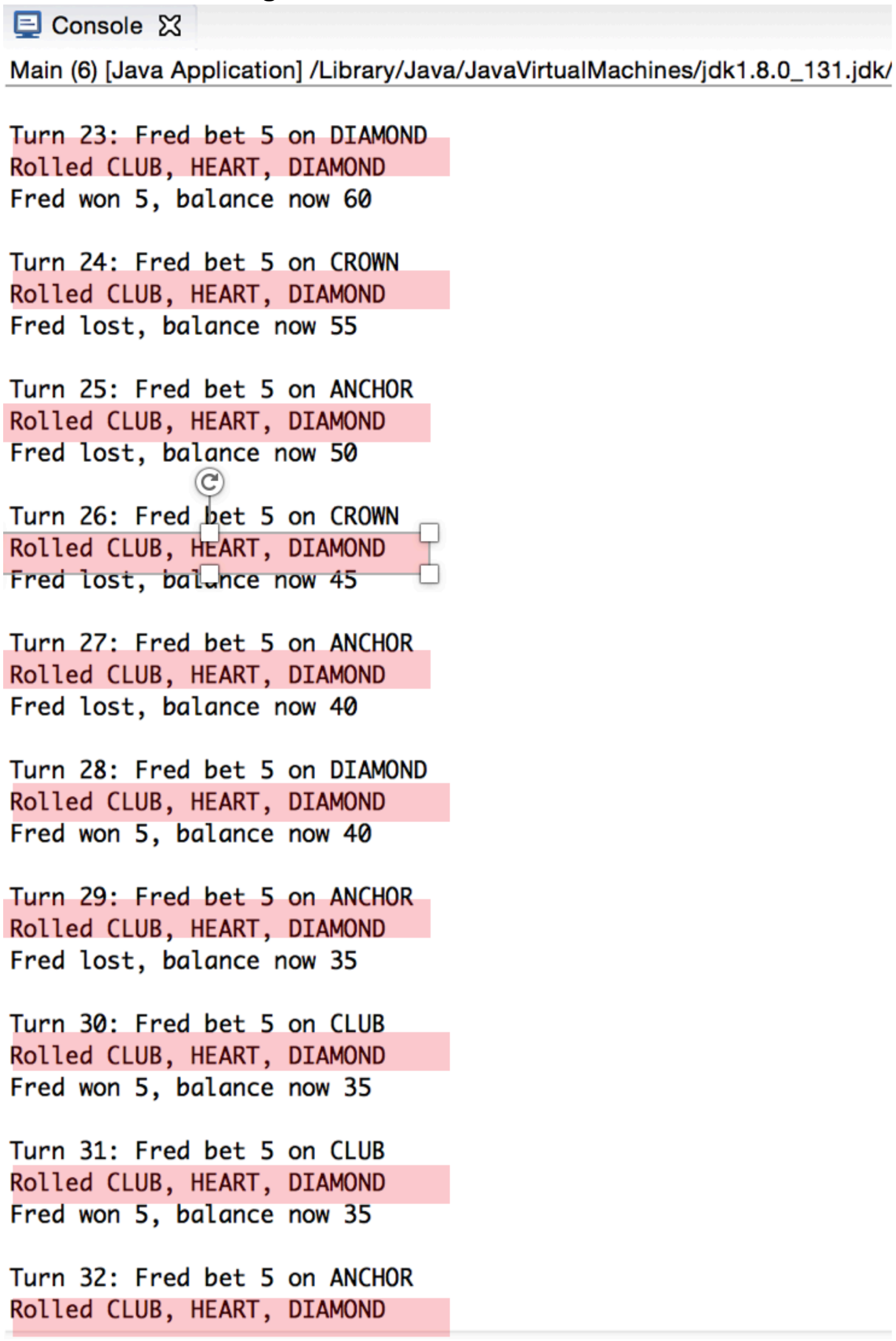


```java
package com.company;
public class Dice {

    private DiceValue value;

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

    public DiceValue getValue() {

        return value;
    }

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

    public String toString() {
        return value.toString();
    }
}

```

Fig: Screenshot highlighting the source of bug.

Before Output:

The following screenshot shows same output of the dice roll for every round of the game which is incorrect as the result of the bug in the program. The dice roll value doesn't change at all after the first round of dice roll.

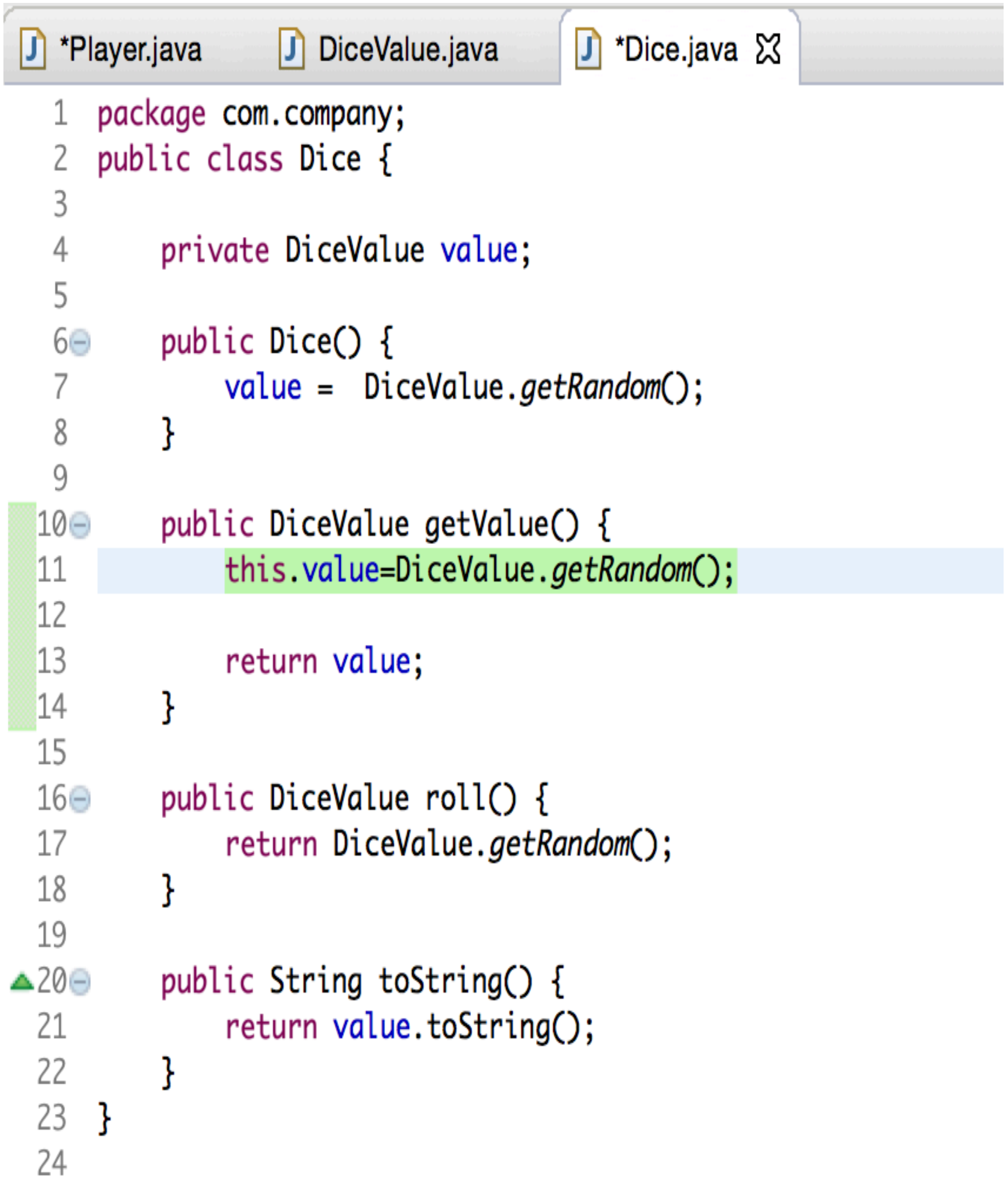


```
Console

Main (6) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/

Turn 23: Fred bet 5 on DIAMOND
Rolled CLUB, HEART, DIAMOND
Fred won 5, balance now 60

Turn 24: Fred bet 5 on CROWN
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 55

Turn 25: Fred bet 5 on ANCHOR
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 50

Turn 26: Fred bet 5 on CROWN
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 45

Turn 27: Fred bet 5 on ANCHOR
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 40

Turn 28: Fred bet 5 on DIAMOND
Rolled CLUB, HEART, DIAMOND
Fred won 5, balance now 40

Turn 29: Fred bet 5 on ANCHOR
Rolled CLUB, HEART, DIAMOND
Fred lost, balance now 35

Turn 30: Fred bet 5 on CLUB
Rolled CLUB, HEART, DIAMOND
Fred won 5, balance now 35

Turn 31: Fred bet 5 on CLUB
Rolled CLUB, HEART, DIAMOND
Fred won 5, balance now 35

Turn 32: Fred bet 5 on ANCHOR
Rolled CLUB, HEART, DIAMOND
```

Fig: Screenshot showing incorrect output as a result of bug.

After Debugging Code:

The following screenshot shows the code that fixes the bug. As the DiceValue getValue() was returning the same value of dice, now I have made it to random so it returns the random value.



*Player.java DiceValue.java *Dice.java

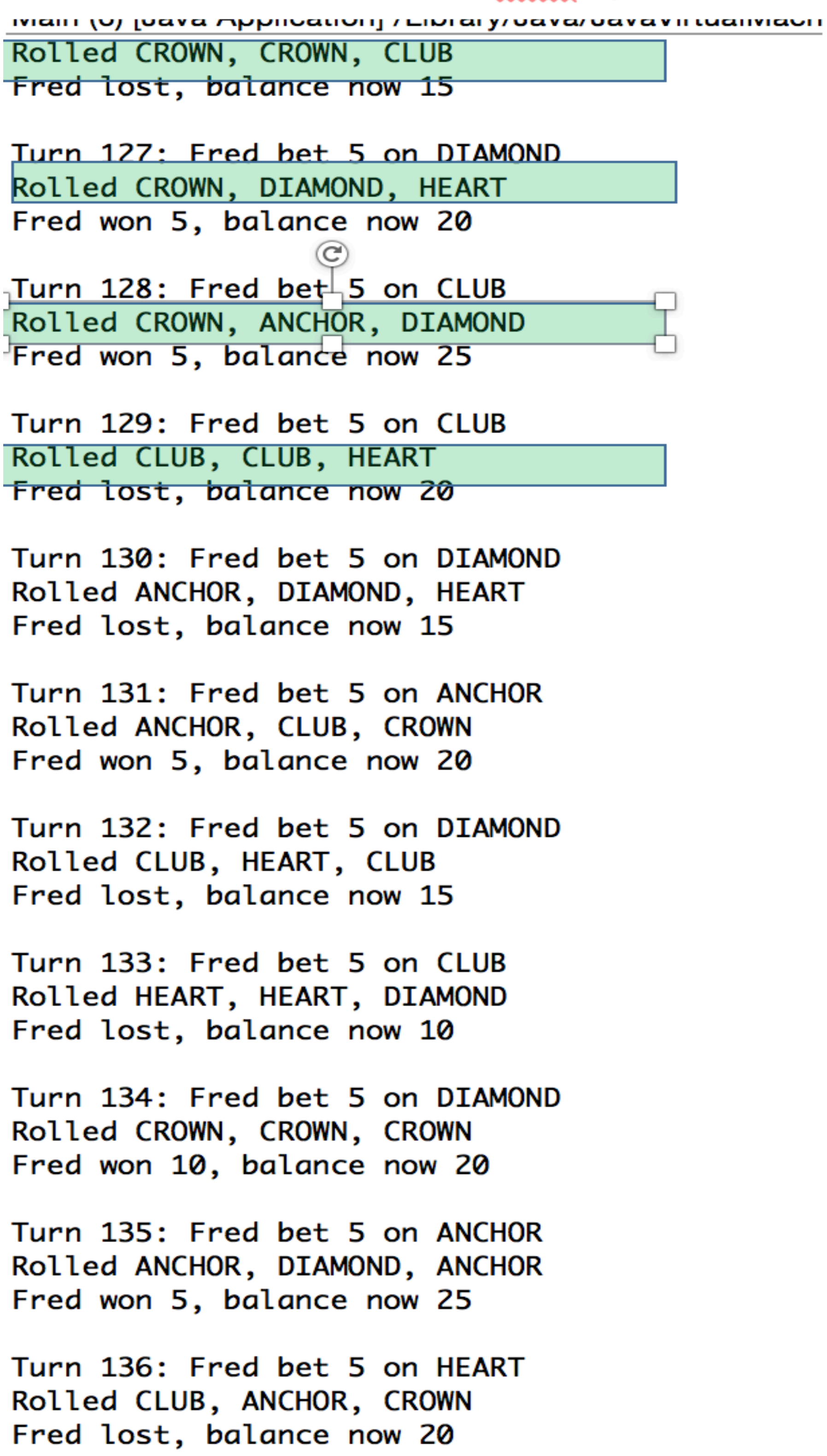
```
package com.company;
public class Dice {

    private DiceValue value;

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

    public DiceValue getValue() {
        this.value=DiceValue.getRandom();

        return value;
    }

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

    public String toString() {
        return value.toString();
    }
}

```

Fig: Screenshot highlighting the fixed code for elimination of bug.

After Debugging Output:

The following screenshot shows the different output of the dice roll for every round of the game which is correct as the result of fixing the bug in the program. The dice roll value does change at all round of dice roll.

Before' and 'after' screen shots identifying the origin of the bug
Bijaya Raj Basnet



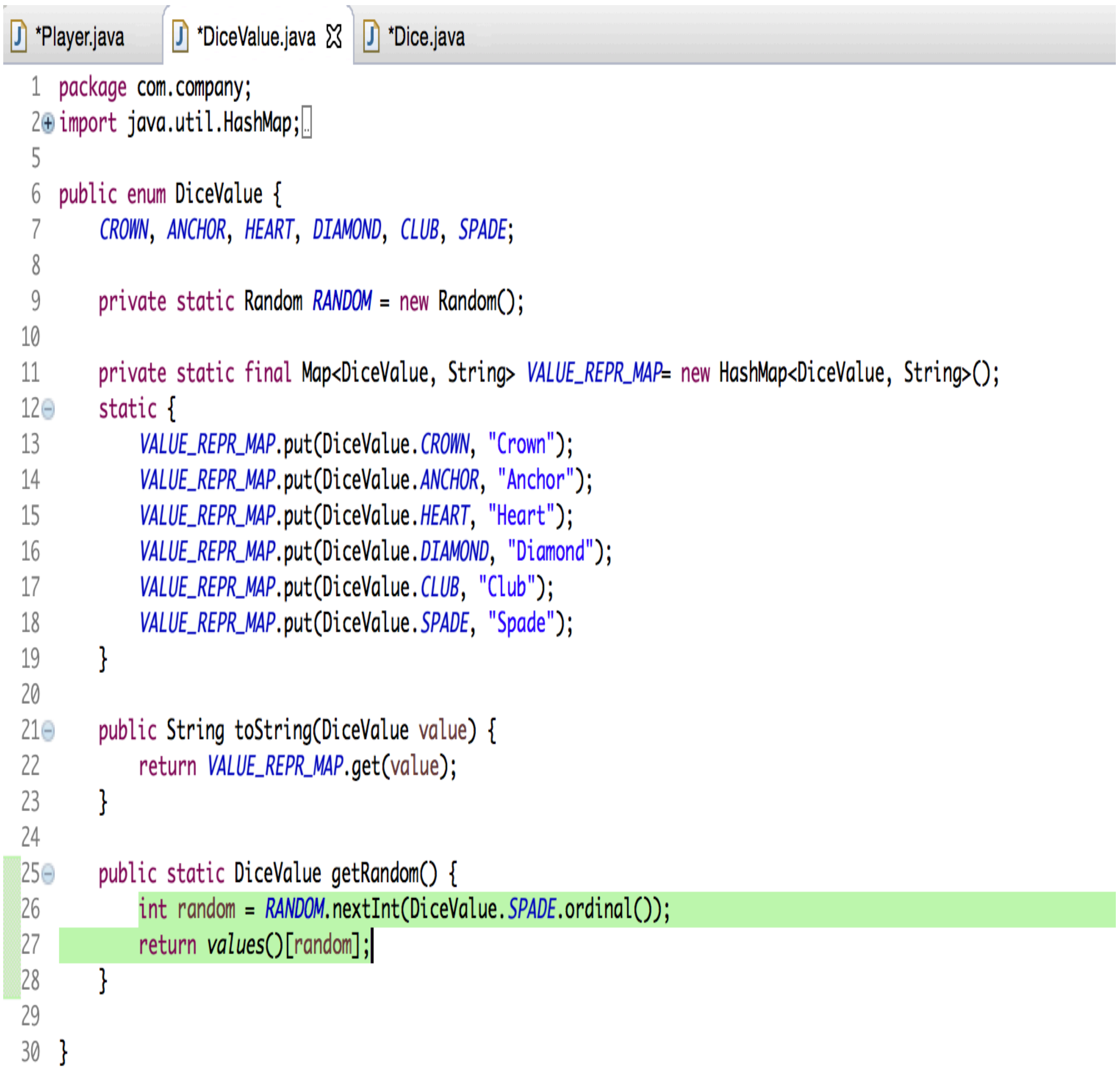
```
Main (0) [Java Application] /Library/Java/JavaVirtualMach
Rolled CROWN, CROWN, CLUB
Fred lost, balance now 15

Turn 127: Fred bet 5 on DIAMOND
Rolled CROWN, DIAMOND, HEART
Fred won 5, balance now 20

Turn 128: Fred bet 5 on CLUB
Rolled CROWN, ANCHOR, DIAMOND
Fred won 5, balance now 25

Turn 129: Fred bet 5 on CLUB
Rolled CLUB, CLUB, HEART
Fred lost, balance now 20

Turn 130: Fred bet 5 on DIAMOND
Rolled ANCHOR, DIAMOND, HEART
Fred lost, balance now 15

Turn 131: Fred bet 5 on ANCHOR
Rolled ANCHOR, CLUB, CROWN
Fred won 5, balance now 20

Turn 132: Fred bet 5 on DIAMOND
Rolled CLUB, HEART, CLUB
Fred lost, balance now 15

Turn 133: Fred bet 5 on CLUB
Rolled HEART, HEART, DIAMOND
Fred lost, balance now 10

Turn 134: Fred bet 5 on DIAMOND
Rolled CROWN, CROWN, CROWN
Fred won 10, balance now 20

Turn 135: Fred bet 5 on ANCHOR
Rolled ANCHOR, DIAMOND, ANCHOR
Fred won 5, balance now 25

Turn 136: Fred bet 5 on HEART
Rolled CLUB, ANCHOR, CROWN
Fred lost, balance now 20
```

Fig: Screenshot showing correct output

Bug 5: “SPADE” doesn’t appear in roll.

Before Code:

The following Screenshot shows the highlighted portion of the code resulting in the bug with no “spade” appear in dice roll. The bug is present in DiceValue.java class , DiceValue getRandon() method. The random assigned in this method skips the SPADE on every occasion of the roll that results in the bug.



*Player.java | *DiceValue.java | *Dice.java

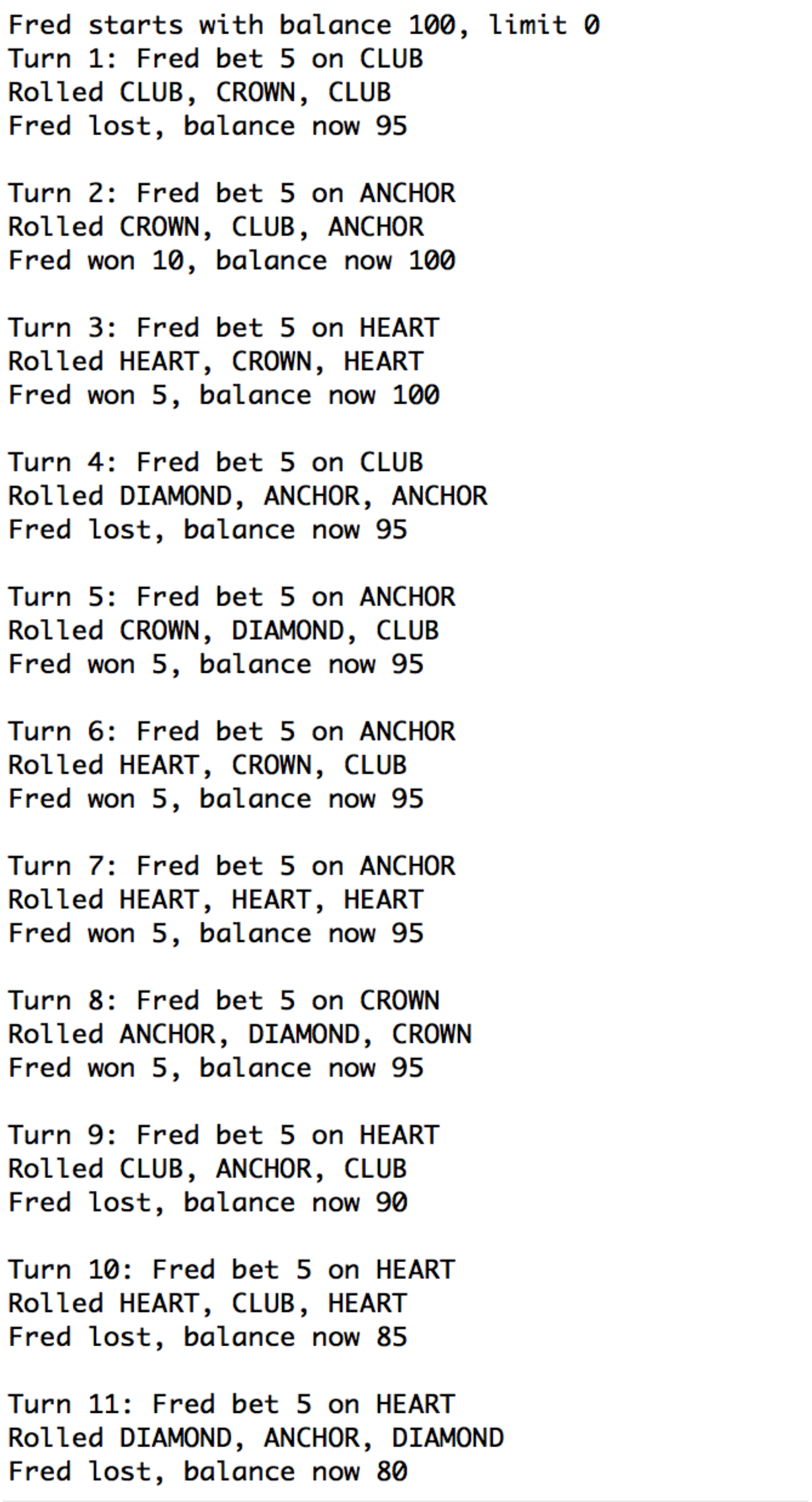
```java
package com.company;
import java.util.HashMap;

public enum DiceValue {
    CROWN, ANCHOR, HEART, DIAMOND, CLUB, SPADE;

    private static Random RANDOM = new Random();

    private static final Map<DiceValue, String> VALUE_REPR_MAP= new HashMap<DiceValue, String>();
    static {
        VALUE_REPR_MAP.put(DiceValue.CROWN, "Crown");
        VALUE_REPR_MAP.put(DiceValue.ANCHOR, "Anchor");
        VALUE_REPR_MAP.put(DiceValue.HEART, "Heart");
        VALUE_REPR_MAP.put(DiceValue.DIAMOND, "Diamond");
        VALUE_REPR_MAP.put(DiceValue.CLUB, "Club");
        VALUE_REPR_MAP.put(DiceValue.SPADE, "Spade");
    }

    public String toString(DiceValue value) {
        return VALUE_REPR_MAP.get(value);
    }

    public static DiceValue getRandom() {
        int random = RANDOM.nextInt(DiceValue.SPADE.ordinal());
        return values()[random];
    }

}
```

Fig: Screenshot highlighting the source of bug.

Before Output:

The following output shows no occurrence of “Spade” as the result of bug. Looking into all the rounds of play there is not even a single occurrence of “spade” which is incorrect.

Before' and 'after' screen shots identifying the origin of the bug
Bijaya Raj Basnet

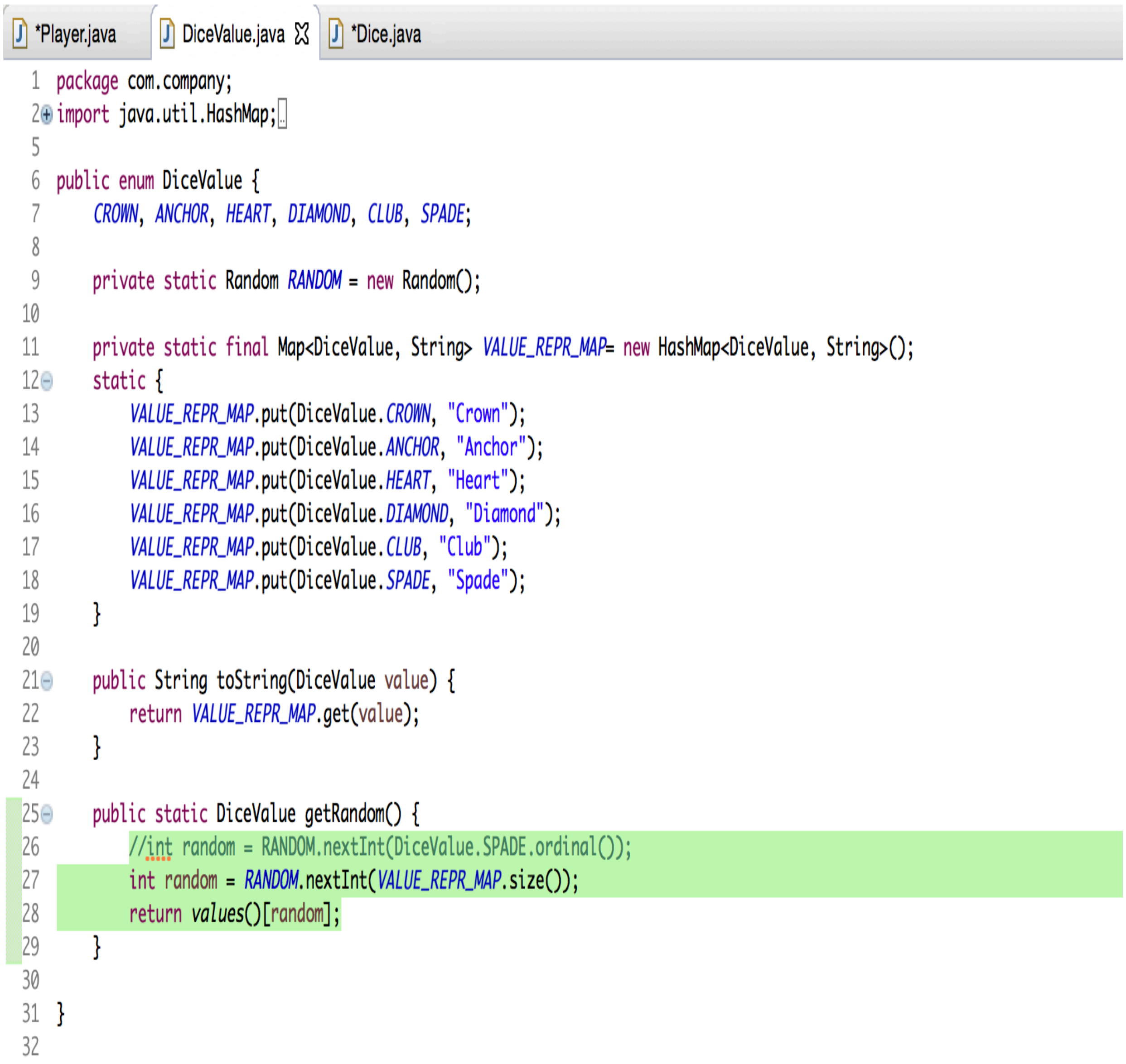


```
Fred starts with balance 100, limit 0
Turn 1: Fred bet 5 on CLUB
Rolled CLUB, CROWN, CLUB
Fred lost, balance now 95

Turn 2: Fred bet 5 on ANCHOR
Rolled CROWN, CLUB, ANCHOR
Fred won 10, balance now 100

Turn 3: Fred bet 5 on HEART
Rolled HEART, CROWN, HEART
Fred won 5, balance now 100

Turn 4: Fred bet 5 on CLUB
Rolled DIAMOND, ANCHOR, ANCHOR
Fred lost, balance now 95

Turn 5: Fred bet 5 on ANCHOR
Rolled CROWN, DIAMOND, CLUB
Fred won 5, balance now 95

Turn 6: Fred bet 5 on ANCHOR
Rolled HEART, CROWN, CLUB
Fred won 5, balance now 95

Turn 7: Fred bet 5 on ANCHOR
Rolled HEART, HEART, HEART
Fred won 5, balance now 95

Turn 8: Fred bet 5 on CROWN
Rolled ANCHOR, DIAMOND, CROWN
Fred won 5, balance now 95

Turn 9: Fred bet 5 on HEART
Rolled CLUB, ANCHOR, CLUB
Fred lost, balance now 90

Turn 10: Fred bet 5 on HEART
Rolled HEART, CLUB, HEART
Fred lost, balance now 85

Turn 11: Fred bet 5 on HEART
Rolled DIAMOND, ANCHOR, DIAMOND
Fred lost, balance now 80
```

Fig: Screenshot showing no “spade” in output as a result of bug.

After debugging code:

The following screenshot shows the highlighted portion of the code that has been debugged to eliminate the "**spade doesn't occurs in dice roll"** bug. I have simply changed the code so that random occurs as per the size of the map that holds the value of the dice rather than the dice Value that was skipping SPADE.



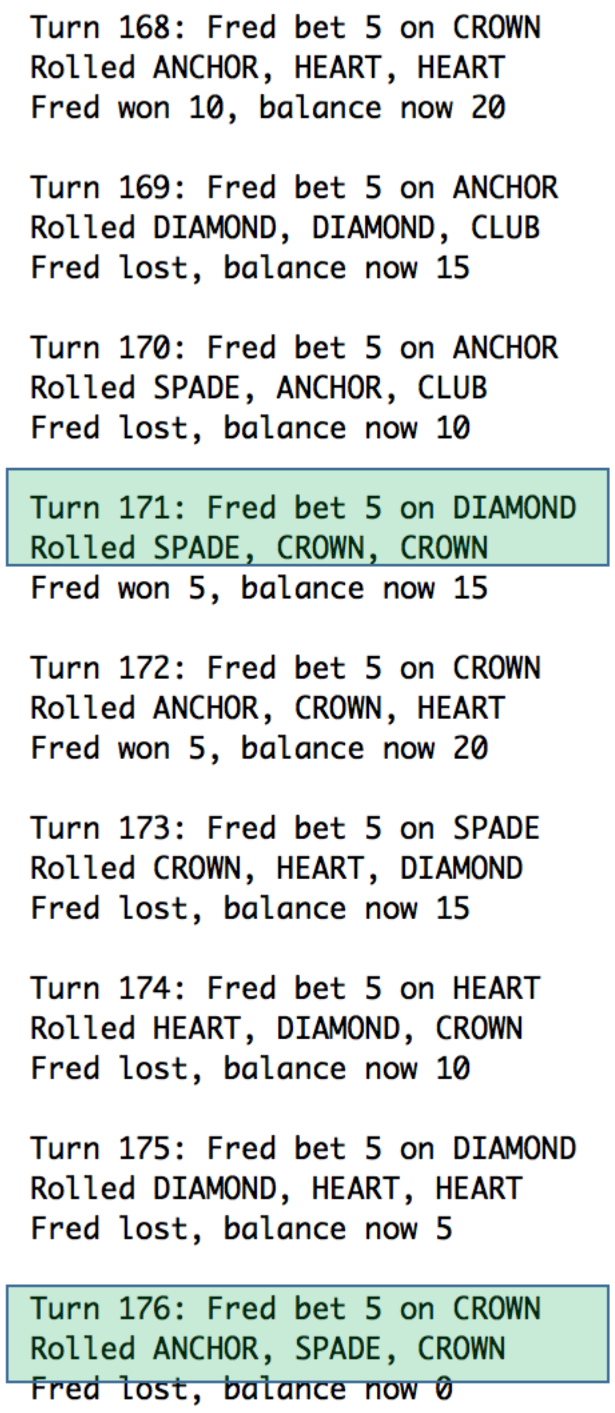
```
package com.company;
import java.util.HashMap;

public enum DiceValue {
    CROWN, ANCHOR, HEART, DIAMOND, CLUB, SPADE;

    private static Random RANDOM = new Random();

    private static final Map<DiceValue, String> VALUE_REPR_MAP= new HashMap<DiceValue, String>();
    static {
        VALUE_REPR_MAP.put(DiceValue.CROWN, "Crown");
        VALUE_REPR_MAP.put(DiceValue.ANCHOR, "Anchor");
        VALUE_REPR_MAP.put(DiceValue.HEART, "Heart");
        VALUE_REPR_MAP.put(DiceValue.DIAMOND, "Diamond");
        VALUE_REPR_MAP.put(DiceValue.CLUB, "Club");
        VALUE_REPR_MAP.put(DiceValue.SPADE, "Spade");
    }

    public String toString(DiceValue value) {
        return VALUE_REPR_MAP.get(value);
    }

    public static DiceValue getRandom() {
        //int random = RANDOM.nextInt(DiceValue.SPADE.ordinal());
        int random = RANDOM.nextInt(VALUE_REPR_MAP.size());
        return values()[random];
    }

}
```

Fig: Screenshot highlighting the fixed code for elimination of bug.

After Debugging Output:

The below screenshot shows the output after fixing the bug. The output clearly shows the occurrence of the "spade" in the dice roll values.

Before' and 'after' screen shots identifying the origin of the bug
Bijaya Raj Basnet



```
Turn 168: Fred bet 5 on CROWN
Rolled ANCHOR, HEART, HEART
Fred won 10, balance now 20

Turn 169: Fred bet 5 on ANCHOR
Rolled DIAMOND, DIAMOND, CLUB
Fred lost, balance now 15

Turn 170: Fred bet 5 on ANCHOR
Rolled SPADE, ANCHOR, CLUB
Fred lost, balance now 10

Turn 171: Fred bet 5 on DIAMOND
Rolled SPADE, CROWN, CROWN
Fred won 5, balance now 15

Turn 172: Fred bet 5 on CROWN
Rolled ANCHOR, CROWN, HEART
Fred won 5, balance now 20

Turn 173: Fred bet 5 on SPADE
Rolled CROWN, HEART, DIAMOND
Fred lost, balance now 15

Turn 174: Fred bet 5 on HEART
Rolled HEART, DIAMOND, CROWN
Fred lost, balance now 10

Turn 175: Fred bet 5 on DIAMOND
Rolled DIAMOND, HEART, HEART
Fred lost, balance now 5

Turn 176: Fred bet 5 on CROWN
Rolled ANCHOR, SPADE, CROWN
Fred lost, balance now 0
```

Fig: Screenshot showing the occurrence of “spade” in dice roll