Bug Reporting and Tracing Summaries

Refer to program output: Sample1.txt (“sample 1”), Sample2.txt (“sample 2”), and Sample3.txt (“sample 3”).

Bugs

[1. One match pays out 0 instead of the expected equal to the bet 2](#_Toc495347304)

[Summary 2](#_Toc495347305)

[Simplification 2](#_Toc495347306)

[Replication 2](#_Toc495347307)

[Tracing 3](#_Toc495347308)

[Resolution 3](#_Toc495347309)

[2. Two matches pay out 1X bet instead of the expected 2X bet 3](#_Toc495347310)

[Summary 3](#_Toc495347311)

[Simplification 3](#_Toc495347312)

[Replication 3](#_Toc495347313)

[Tracing 4](#_Toc495347314)

[Resolution 4](#_Toc495347315)

[3. Three matches pays out 2X bet instead of the expected 3X bet 4](#_Toc495347316)

[Summary 4](#_Toc495347317)

[Simplification 4](#_Toc495347318)

[Replication 5](#_Toc495347319)

[Tracing 5](#_Toc495347320)

[Resolution 5](#_Toc495347321)

[4. All rolls are the same in each run through 5](#_Toc495347322)

[Summary 5](#_Toc495347323)

[Simplification 5](#_Toc495347324)

[Replication 5](#_Toc495347325)

[Tracing 6](#_Toc495347326)

[Resolution 6](#_Toc495347327)

[5. Spades are never rolled 6](#_Toc495347328)

[Summary 6](#_Toc495347329)

[Simplification 6](#_Toc495347330)

[Tracing 6](#_Toc495347331)

[Resolution 6](#_Toc495347332)

[6. Player never guesses “Spade” 6](#_Toc495347333)

[Summary 6](#_Toc495347334)

[Simplification 6](#_Toc495347335)

[Tracing 6](#_Toc495347336)

[Resolution 6](#_Toc495347337)

[7. Odds of game incorrect 7](#_Toc495347338)

[Summary 7](#_Toc495347339)

[Simplification 7](#_Toc495347340)

[Tracing 7](#_Toc495347341)

[8. Betting limit unable to be reached 7](#_Toc495347342)

[Summary 7](#_Toc495347343)

[Simplification 7](#_Toc495347344)

[Tracing 7](#_Toc495347345)

[Resolution 7](#_Toc495347346)

# One match pays out 0 instead of the expected equal to the bet

## Summary

* Game pays out incorrect winnings.
* Initially found through unit testing the Game class.
* Also Bug 1 in initial bug report provided with assignment.

## Simplification

1. Displayed in sample 1 on turn 1:

*Fred starts with balance 100, limit 0*

*Turn 1: Fred bet 5 on CROWN*

*Rolled HEART, HEART, CROWN*

*Fred won 5, balance now 100*

Fred got one match, so should win $5. He started with $100, so should have $105. But he only has $100 still.

1. Fails GameTest unit testing in test TestPlayRoundOneMatch.

## Replication

It isn’t really feasible to make a dedicated replication program for this bug. It would require too much manipulation of main, which has the risk of introducing other defects (for example, maybe we would have condition to only print to screen when there was exactly one match). Other bugs have replication because it is just a matter of turning messages on or off to see the results more clearly, or just adding up the wins and losses over many runs of the program. This isn’t the case here.

However, we can run the Game class unit test (GameTest.java) to simulate one match and verify that the player gets the wrong amount in their balance as a result.

We can also have a structured set of steps for observing the bug through inspection of normal results.

Please see Bug 1 UAT Replication and Simplification.docx and Bug 1 UAT Replication (Unit Test).docx for details.

## Tracing

* Same cause as bugs 2 and 3.
* The value of the dice is compared successfully to pick and the correct number of matches is calculated.
* The player’s balance correctly increases by the winnings amount after receiveWinnings is executed.
* The balance is incorrect at the time of executing receiveWinnings.
* The moment the balance becomes incorrect is just after takeBet is called. (i.e. to use correct terminology the player’s balance becomes ‘infected’ by takeBet).

All Tracing tested and verified (please see: Bugs 1 2 and 3 Tracing.docx).

## Resolution

Moving player.takeBet from the beginning of playRound to the end, and only executing it if the player makes one or more matches.

# Two matches pay out 1X bet instead of the expected 2X bet

## Summary

* Game pays out incorrect winnings.
* Initially found through unit testing the Game class.

## Simplification

1. Displayed in sample 1 on turn 1:

Fred lost, balance now 95

Turn 3: Fred bet 5 on HEART

Rolled HEART, HEART, CROWN

Fred won 10, balance now 100

Since Fred got two matches, he wins $10. He had $95 before, so, he should have $105 now. But he only has $100.

1. Fails GameTest unit testing in test TestPlayRoundTwoMatches.

## Replication

It isn’t really feasible to make a dedicated replication program for this bug. It would require too much manipulation of main, which has the risk of introducing other defects (for example, maybe we would have condition to only print to screen when there were exactly two matches). Other bugs have replication because it is just a matter of turning messages on or off to see the results more clearly, or just adding up the wins and losses over many runs of the program. This isn’t the case here.

However, we can run the Game class unit test (GameTest.java) to simulate two matches and verify that the player gets the wrong amount in their balance as a result.

We can also have a structured set of steps for observing the bug through inspection of normal results.

Please see Bug 2 UAT Replication and Simplification.docx and Bug 2 UAT Replication (Unit Test).docx for details.

## Tracing

* Same cause as bugs 1 and 3.
* The value of the dice is compared successfully to pick and the correct number of matches is calculated.
* The player’s balance correctly increases by the winnings amount after receiveWinnings is executed.
* The balance is incorrect at the time of executing receiveWinnings.
* The moment the balance becomes incorrect is just after takeBet is called. (i.e. to use correct terminology the player’s balance becomes ‘infected’ by takeBet).

All Tracing tested and verified (please see: Bugs 1 2 and 3 Tracing.docx).

## Resolution

Moving player.takeBet from the beginning of playRound to the end, and only executing it if the player makes one or more matches.

# 3. Three matches pays out 2X bet instead of the expected 3X bet

## Summary

* Game pays out incorrect winnings.
* Initially found through unit testing the Game class.

## Simplification

1. Displayed in Bugs123InspectionUATReplication.txt

Fred lost, balance now 30

Turn 49: Fred bet 5 on CLUB

Rolled CLUB, CLUB, CLUB

Fred won 15, balance now 40

Since Fred got three matches, he wins $15. He had $30 before, so, he should have $45 now. But he only has $40.

1. Fails GameTest unit testing in test TestPlayRoundThreeMatches.

## Replication

It isn’t really feasible to make a dedicated replication program for this bug. It would require too much manipulation of main, which has the risk of introducing other defects (for example, maybe we would have condition to only print to screen when there were exactly three matches). Other bugs have replication because it is just a matter of turning messages on or off to see the results more clearly, or just adding up the wins and losses over many runs of the program. This isn’t the case here.

However, we can run the Game class unit test (GameTest.java) to simulate three matches and verify that the player gets the wrong amount in their balance as a result.

We can also have a structured set of steps for observing the bug through inspection of normal results.

Please see Bug 3 UAT Replication (Inspection).docx and Bug 3 UAT Replication (Unit Test).docx for details.

## Tracing

* Same cause as bugs 1 and 2.
* The value of the dice is compared successfully to pick and the correct number of matches is calculated.
* The player’s balance correctly increases by the winnings amount after receiveWinnings is executed.
* The balance is incorrect at the time of executing receiveWinnings.
* The moment the balance becomes incorrect is just after takeBet is called. (i.e. to use correct terminology the player’s balance becomes ‘infected’ by takeBet).

All Tracing tested and verified (please see: Bugs 1 2 and 3 Tracing.docx).

## Resolution

Moving player.takeBet from the beginning of playRound to the end, and only executing it if the player makes one or more matches.

# All rolls are the same in each run through

## Summary

* In each run through, the rolls are all the same.
* Initially found by running the program and observing the results.

## Simplification

1. In sample 1, all the rolls are Heart, Heart, Crown (38 rolls).
2. In sample 2, all the rolls are Diamond, Diamond, Club (69 rolls).
3. In sample 3, the rolls are all Anchor, Heart, Anchor (45 rolls).

## Replication

Please see Bug 4 UAT Replication.docx

“Bug4Replication.java” executes **Main**’s **main** but takes out all the peripheral comments, leaving just the rolls so you can see more at a glance, and not see anything that isn’t important.

## Tracing

1. **Dice** are created only once per run of the program, and then reused for each **Game**.
2. The **value** is invariant over the life of any particular instance of **Dice**.
3. The **value** of the **Dice** is what is used as each roll & compared to the pick to determine if the player wins or not.

All Tracing tested and found to be verified (please see Bug 4 Tracing.docx for details).

## Resolution

The **roll** method in the **Dice** class was edited to save the result to value. This successfully resolved the bug.

# Spades are never rolled

## Summary

* Spades are never rolled during the game.
* Initially found during unit testing of DiceValue (after 100 rolls, Spades are not produced using the function getRandom).

## Simplification

1. Fails DiceValueTest unit testing in test TestGetRandomsProducesSpade
2. No spades rolled in same 1, sample 2, or sample 3.

## Tracing

1. getRandom can’t ever return spades
2. Bugs 5 and 6 are caused by the same programming error

All Tracing tested and verified. (Please see: Bugs 5 6 and 7 Tracing.docx)

## Resolution

Adding +1 to the argument of RANDOM.nextInt in DiceValue’s getRandom.

# Player never guesses “Spade”

## Summary

* Player never picks “Spade” to bet on.

## Simplification

1. Player never picks “Spade” to bet on in sample 1, sample 2, or sample 3.

## Tracing

1. getRandom can’t ever return spades
2. Bugs 5 and 6 are caused by the same programming error

All Tracing tested and verified. (Please see: Bugs 5 6 and 7 Tracing.docx)

## Resolution

Adding +1 to the argument of RANDOM.nextInt in DiceValue’s getRandom

# Odds of game incorrect

## Summary

Odds appear to be 48.8% win rate instead of the expected 42.0%.

## Simplification

1. After running Bug7Replication.java

## Tracing

1. Same cause as bugs 5 and 6.

Hypothesis tested and verified. (Please see: Bugs 5 6 and 7 Tracing.docx)

# Betting limit unable to be reached

## Summary

The game is meant to continue until the player reaches double his starting balance or his betting limit. But the game ends when he still has $5 left.

## Simplification

* In Sample1.txt:

79 turns later.

End Game 0: Fred now has balance 5

* In Sample2.txt:

53 turns later.

End Game 0: Fred now has balance 5

* In Sample3.txt:

58 turns later.

End Game 0: Fred now has balance 5

## Tracing

1. When the amount to be bet would reduce the balance to the limit if lost, the Game is unable to be played (incorrectly).
2. When the amount to be bet would reduce the balance below the limit if lost, the Game is unable to be played (correctly).
3. When the amount to be bet would reduce the balance to above the limit, the Game can be played (correctly).

All Tracing tested and verified. (please see Bug 8 Tracing.docx)

## Resolution

The cause of the bug is that it doesn’t allow the balance – the bet to equal limit. Therefore, we can add an equals sign in the method balanceExceedsLimitBy.