



AARHUS UNIVERSITET

Software Engineering and Architecture

An informal guide to
Debugging

Definition: **Failure**

A failure is a situation in which the behavior of the executing software deviates from what is expected.

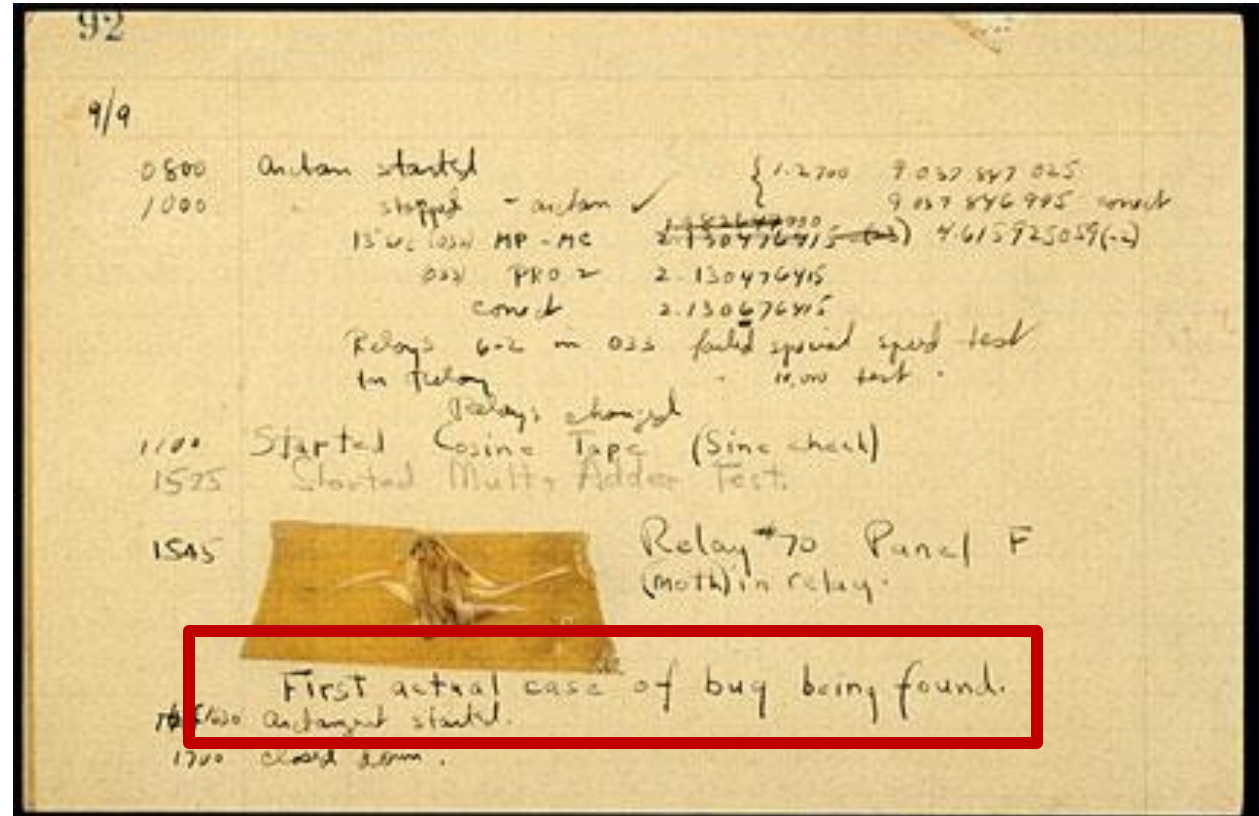
- Why are there failures in our running programs?

Definition: **Defect**

A defect is the algorithmic cause of a failure: some code logic that is incorrectly implemented.

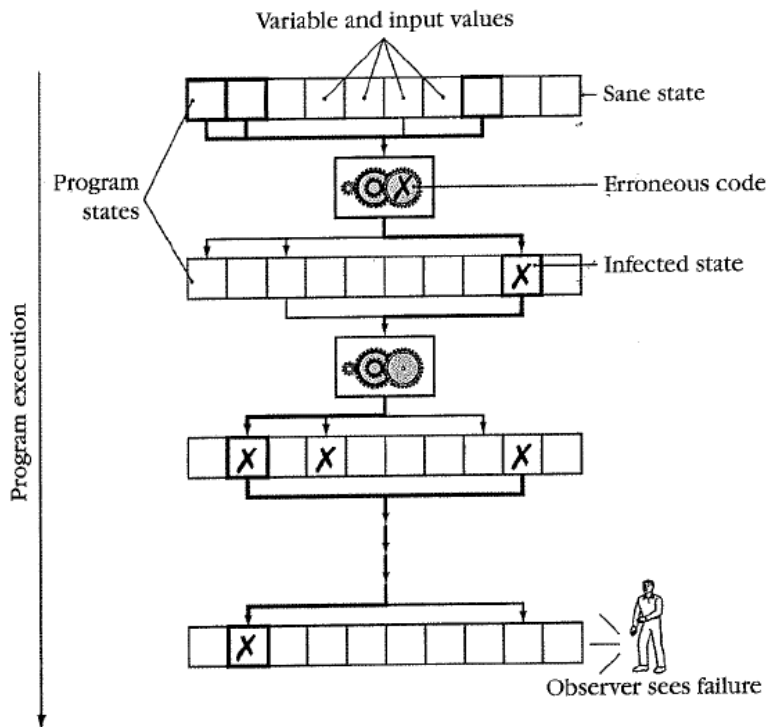
- Or 'Bug'

Worlds First Bug



Debugging

- Some part of ‘input space’ trigger the Defect and leads to a failure
 - Software state is incorrect
- As in
 - The output is incorrect
 - “does wrong things”
 - System hangs
 - System crashes

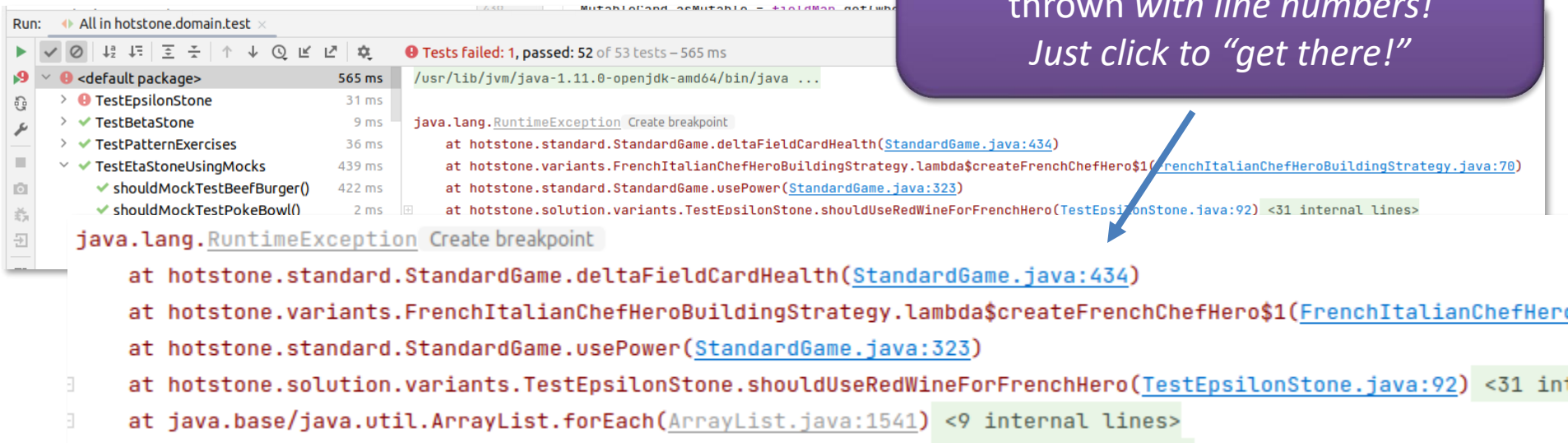


- Debugging is phased
 - **Reproduce the defect**
 - Find the exact input space/situation/context in which it appears
 - “Ah, the reverse thrust of Airbus 320 cannot be engaged to stop the plane on the runway after landing, if one set of wheels hit the ground very smoothly...”
 - » [This is an actual bug, that stranded a plane outside the runway!]
 - **Find the ‘infection origin’**
 - Find the code that misbehaves and what state triggers it
 - Understand the code ‘leading up to the defect’
 - Why was the state that, leading up to the defect
 - **Write an automated test that reproduce the bug**
 - **Fix it – rewrite that defective code**

Stack Traces

- Java + IntelliJ is a major help for modern day development as you get a *Stack trace* for exceptions!
 - “the code lead up to the failure”
- Example*

Stack trace: The sequence of calls leading up to the exception begin thrown with *line numbers!*
Just click to “get there!”



Run: All in hotstone.domain.test x

Tests failed: 1, passed: 52 of 53 tests – 565 ms

<default package> 565 ms

TestEpsilonStone 31 ms

TestBetaStone 9 ms

TestPatternExercises 36 ms

TestEtaStoneUsingMocks 439 ms

shouldMockTestBeefBurger() 422 ms

shouldMockTestPokeBowl() 2 ms

java.lang.RuntimeException Create breakpoint

at hotstone.standard.StandardGame.deltaFieldCardHealth(StandardGame.java:434)

at hotstone.variants.FrenchItalianChefHeroBuildingStrategy.lambda\$createFrenchChefHero\$1(FrenchItalianChefHeroBuildingStrategy.java:70)

at hotstone.standard.StandardGame.usePower(StandardGame.java:323)

at hotstone.solution.variants.TestEpsilonStone.shouldUseRedWineForFrenchHero(TestEpsilonStone.java:92) <31 internal lines>

java.lang.RuntimeException Create breakpoint

at hotstone.standard.StandardGame.deltaFieldCardHealth(StandardGame.java:434)

at hotstone.variants.FrenchItalianChefHeroBuildingStrategy.lambda\$createFrenchChefHero\$1(FrenchItalianChefHeroBuildingStrategy.java:70)

at hotstone.standard.StandardGame.usePower(StandardGame.java:323)

at hotstone.solution.variants.TestEpsilonStone.shouldUseRedWineForFrenchHero(TestEpsilonStone.java:92) <31 internal lines>

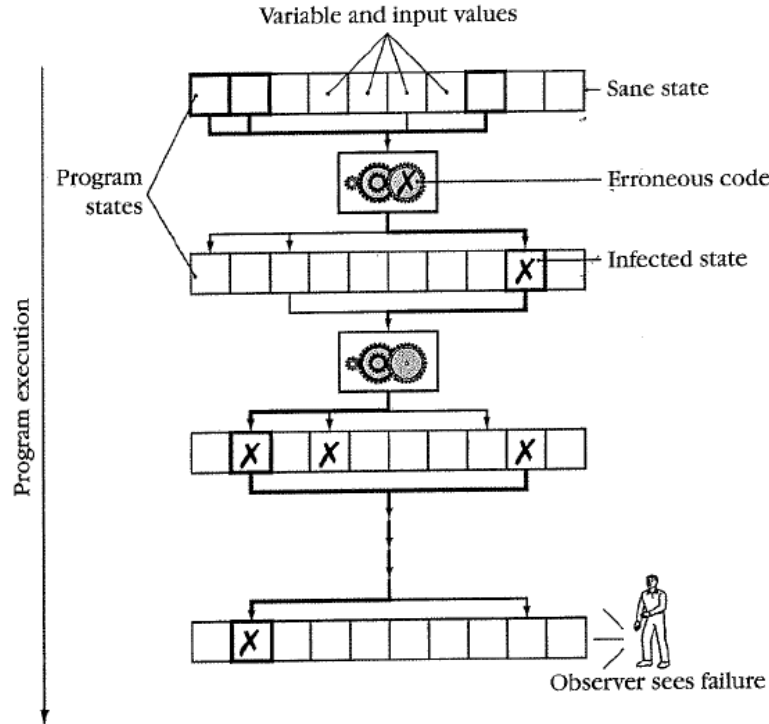
at java.base/java.util.ArrayList.forEach(ArrayList.java:1541) <9 internal lines>

Stack Traces

help for modern day
Stack trace for exceptions!

ure”

Stack trace: The sequence of calls leading up to the exception begin thrown with line numbers!
Just click to “get there!”



in/java ...

```

ltaFieldCardHealth(StandardGame.java:434)
efHeroBuildingStrategy.lambda$createFrenchChefHero$1(FrenchItalianChefHeroBuildingStrategy.java:70)
ePower(StandardGame.java:323)
silonStone.shouldUseRedWineForFrenchHero(TestEpsilonStone.java:92) <31 internal lines>

rdHealth(StandardGame.java:434)
dingStrategy.lambda$createFrenchChefHero$1(FrenchItalianChefHero
ndardGame.java:323)

at java.base/java.util.ArrayList.forEach(ArrayList.java:1541) <9 internal lines>

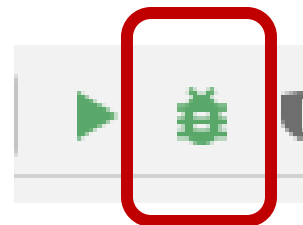
```

The Detective Work

- Sometime the reason for the failure is “a long way” from the place where it manifests itself!
- Sometimes failures can lurk in the dark for years
 - If the ‘triggering state’ is encountered for the first time due to some on-the-surface change in another part of the system...
- Detective work
 - “Ok, failure because x was -1, but why it ever set to -1? It was passed by this method, which was called from this code, which was...”

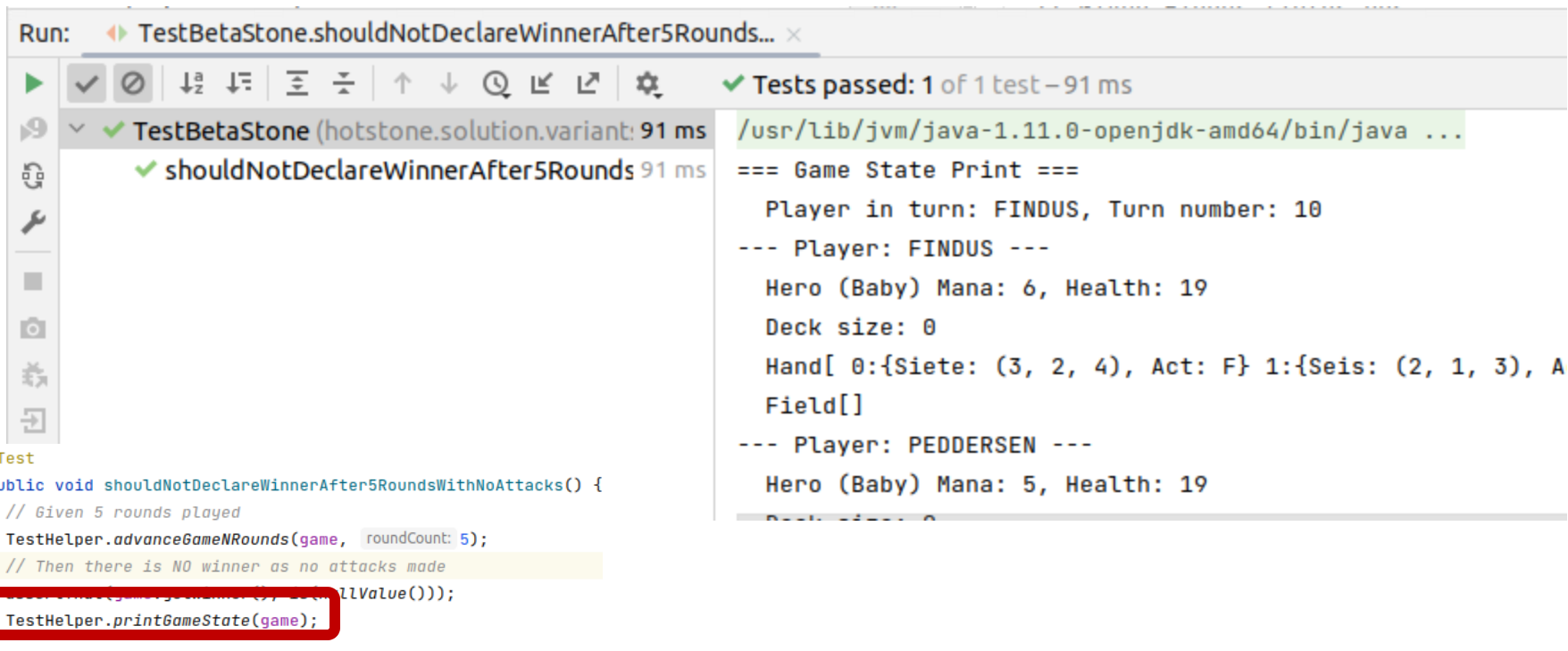
Help The Detective Work

- Often defects are because of *incorrect assumptions*
 - *State space is different from what I expected*
- Either
 - Use the debugger
 - Which has a steep learning curve in itself ☹️
 - Or the old workhorse... **Add print statements**
 - `System.out.println("→ In playCard, index = "`
 - `+ index + ", list size = " + myHand.size());`
- I have not started the debugger in years
 - *Take small steps* saves a lot of debugging!
- But note the → marker, allow search+remove afterwards!



Helpers in HotStone

- `TestHelper.printGameState(game);`



The screenshot shows an IDE window with a test run titled "TestBetaStone.shouldNotDeclareWinnerAfter5Rounds...". The test passed, and the output is displayed in a console window. The output shows the game state printout for the current turn.

```
Run: TestBetaStone.shouldNotDeclareWinnerAfter5Rounds... x
Tests passed: 1 of 1 test - 91 ms

TestBetaStone (hotstone.solution.variant: 91 ms)
  shouldNotDeclareWinnerAfter5Rounds 91 ms

/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java ...
=== Game State Print ===
  Player in turn: FINDUS, Turn number: 10
  --- Player: FINDUS ---
    Hero (Baby) Mana: 6, Health: 19
    Deck size: 0
    Hand[ 0:{Siete: (3, 2, 4), Act: F} 1:{Seis: (2, 1, 3), A
    Field[]
  --- Player: PEDDERSEN ---
    Hero (Baby) Mana: 5, Health: 19
```

The code snippet on the left shows the test method `shouldNotDeclareWinnerAfter5RoundsWithNoAttacks()`. The line `TestHelper.printGameState(game);` is highlighted with a red box.

```
@Test
public void shouldNotDeclareWinnerAfter5RoundsWithNoAttacks() {
    // Given 5 rounds played
    TestHelper.advanceGameNRounds(game, roundCount: 5);
    // Then there is NO winner as no attacks made
    TestHelper.printGameState(game);
}
```

Example

- While developing the GUI for HotStone, I encountered this failure...

```
---> HITTING: OPPONENT_ACTION_BUTTON
---> switching to OppActTool!
--> Enacting mouse down on State=class hotstone.solution.main.OpponentActionTool
==> I am in the OppActTool.
==> I casted - the type is hotstone.recordplayback.MyGameEventReplyDecorator
--> MyGameEventReplyDecorator START
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
    at hotstone.view.core.HotStoneDrawing.refreshField(HotStoneDrawing.java:465)
    at hotstone.view.core.HotStoneDrawing.onCardPlay(HotStoneDrawing.java:327)
    at hotstone.standard.ObserverHandler.lambda$firePlayCard$0(ObserverHandler.java:41)
    at java.base/java.util.ArrayList$ArrayListSpliterator.forEachRemaining(ArrayList.java:1655)
    at java.base/java.util.stream.ReferencePipeline$Head.forEach(ReferencePipeline.java:658)
    at hotstone.standard.ObserverHandler.firePlayCard(ObserverHandler.java:41)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$3(GameEventPlayer.java:75)
```

- (In the 'good old days' I would *not get a stack trace*)

- Identifying the exact type of *failure*
 - Here: some code did 'object.method()' and object == null ☹*

```
---> HITTING: OPPONENT_ACTION_BUTTON
---> switching to OppActTool!
--> Enacting mouse down on State=class hotstone.solution.main.OpponentActionTool
==> I am in the OppActTool.
==> I casted - the type is hotstone.recordplayback.MyGameEventReplyDecorator
-> MyGameEventReplayDecorator START
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
    at hotstone.view.core.HotStoneDrawing.refreshField(HotStoneDrawing.java:465)
    at hotstone.view.core.HotStoneDrawing.onCardPlay(HotStoneDrawing.java:327)
    at hotstone.standard.ObserverHandler.lambda$firePlayCard$0(ObserverHandler.java:41)
    at java.base/java.util.ArrayList$ArrayListSpliterator.forEachRemaining(ArrayList.java:1655)
    at java.base/java.util.stream.ReferencePipeline$Head.forEach(ReferencePipeline.java:658)
    at hotstone.standard.ObserverHandler.firePlayCard(ObserverHandler.java:41)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$3(GameEventPlayer.java:75)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$4(GameEventPlayer.java:85)
```

- It is the *full call graph when the failure occurred*
 - *refreshField* was called by *onCardPlay* called by *FirePlayCard* called by *forEachRemaining* and so on...
 - So the '*object == null*' is within the *refreshField* of *HotStoneDrawing*

```

---> HITTING: OPPONENT_ACTION_BUTTON
---> switching to OppActTool!
--> Enacting mouse down on State=class hotstone.solution.main.OpponentActionTool
==> I am in the OppActTool.
==> I casted - the type is hotstone.recordplayback.MyGameEventReplyDecorator
-> MyGameEventReplayDecorator START
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
    at hotstone.view.core.HotStoneDrawing.refreshField(HotStoneDrawing.java:465)
    at hotstone.view.core.HotStoneDrawing.onCardPlay(HotStoneDrawing.java:327)
    at hotstone.standard.ObserverHandler.lambda$firePlayCard$0(ObserverHandler.java:41)
    at java.base/java.util.ArrayList$ArrayListSpliterator.forEachRemaining(ArrayList.java:1655)
    at java.base/java.util.stream.ReferencePipeline$Head.forEach(ReferencePipeline.java:658)
    at hotstone.standard.ObserverHandler.firePlayCard(ObserverHandler.java:41)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$3(GameEventPlayer.java:75)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$4(GameEventPlayer.java:75)

```

StackTrace

- Moreover, since Java is running in its own execution environment (java virtual machine) it knows a lot about the executing code
 - *I get the exact spot in the code: line 465*

```
---> HITTING: OPPONENT_ACTION_BUTTON
---> switching to OppActTool!
--> Enacting mouse down on State=class hotstone.solution.main.OpponentActionTool
==> I am in the OppActTool.
==> I casted - the type is hotstone.recordplayback.MyGameEventReplyDecorator
-> MyGameEventReplayDecorator START
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException
    at hotstone.view.core.HotStoneDrawing.refreshField(HotStoneDrawing.java:465)
    at hotstone.view.core.HotStoneDrawing.onCardPlay(HotStoneDrawing.java:327)
    at hotstone.standard.ObserverHandler.lambda$firePlayCard$0(ObserverHandler.java:41)
    at java.base/java.util.ArrayList$ArrayListSpliterator.forEachRemaining(ArrayList.java:1655)
    at java.base/java.util.stream.ReferencePipeline$Head.forEach(ReferencePipeline.java:658)
    at hotstone.standard.ObserverHandler.firePlayCard(ObserverHandler.java:41)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$3(GameEventPlayer.java:75)
    at hotstone.recordplayback.GameEventPlayer.lambda$new$4(GameEventPlayer.java:75)
```

Which means I can...

- Go to the source code and review to identify what went wrong...

```
454 public void refreshField(Player who) {  
455     // TODO: Fix magic constants all over the place  
456     int count = 0;  
457     int distance = 150;  
458     int fieldSize = game.getFieldSize(who);  
459     int offset = fieldSize % 2 == 0 ? 0 : distance/2;  
460     int centerX = 600 - offset - (fieldSize/2) * distance;  
461     int yPos = (who == playerShown ? 500 : 300);  
462     for (Card card: game.getField(who)) {  
463         assert card.getOwner() == who;  
464         ActorBaseFigure actor = actorMap.get(card);  
465         actor.setPosition(absoluteX: centerX + distance * count++, yPos);  
466     }  
467 }
```

Exercise

- So what is the **defect** that has lead to the **failure** ?
 - Remember: it was a null pointer exception (null reference)

```
454 public void refreshField(Player who) {  
455     // TODO: Fix magic constants all over the place  
456     int count = 0;  
457     int distance = 150;  
458     int fieldSize = game.getFieldSize(who);  
459     int offset = fieldSize % 2 == 0 ? 0 : distance/2;  
460     int centerX = 600 - offset - (fieldSize/2) * distance;  
461     int yPos = (who == playerShown ? 500 : 300);  
462     for (Card card: game.getField(who)) {  
463         assert card.getOwner() == who;  
464         ActorBaseFigure actor = actorMap.get(card);  
465         actor.setPosition(absoluteX: centerX + distance * count++, yPos);  
466     }  
467 }
```


Lessons Learned

- *Development stops when a failure occurs – immediately*
 - *You start debugging at once. Do not defects pile up!*
- It can be tempting to ‘just get that nice feature working and postpone debugging’
 - **Do Not.** If defects pile up you simply loose focus. **Keep Focus!**
- The almost-finished-sheet-music-editor warstory...

When Defect Found...

- Sometimes you spend hours tracking down that damn defect...
- When you find it: *Heureka!*
- ***But you do not fix it, right?***
 - *What do you do?*
- Exception: pure gui code.