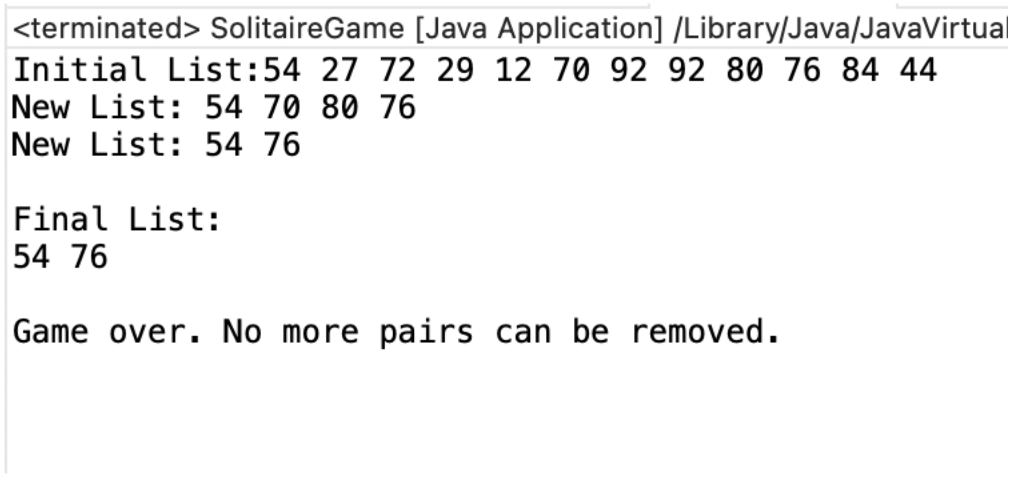
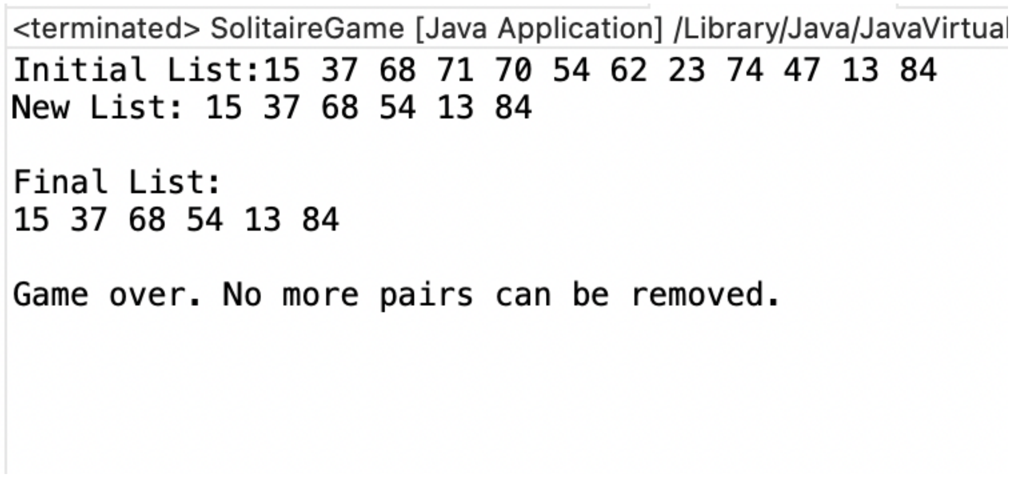
Lab Iterators

Sousanna Chugunova

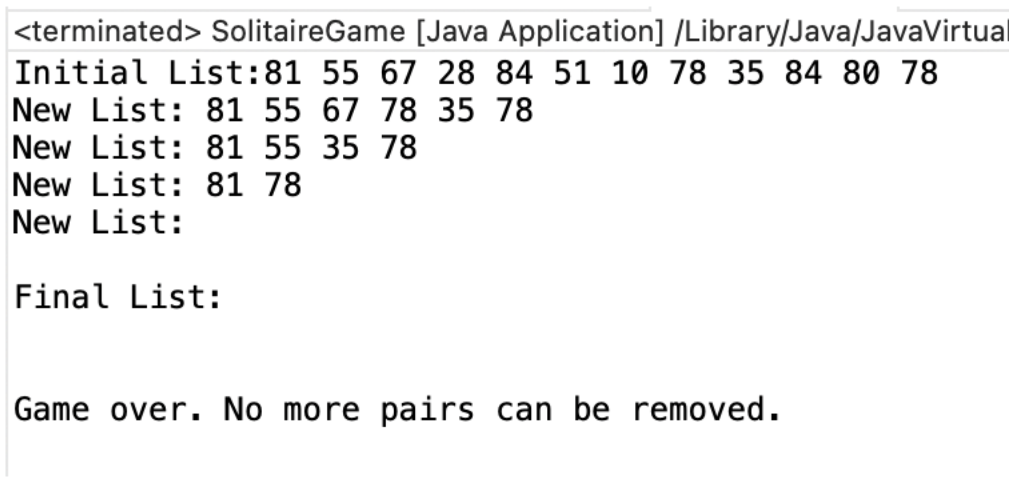
Runs:



```
<terminated> SolitaireGame [Java Application] /Library/Java/JavaVirtual
Initial List:54 27 72 29 12 70 92 92 80 76 84 44
New List: 54 70 80 76
New List: 54 76

Final List:
54 76

Game over. No more pairs can be removed.
```



```
<terminated> SolitaireGame [Java Application] /Library/Java/JavaVirtual
Initial List:15 37 68 71 70 54 62 23 74 47 13 84
New List: 15 37 68 54 13 84

Final List:
15 37 68 54 13 84

Game over. No more pairs can be removed.
```



```
<terminated> SolitaireGame [Java Application] /Library/Java/JavaVirtual
Initial List:81 55 67 28 84 51 10 78 35 84 80 78
New List: 81 55 67 78 35 78
New List: 81 55 35 78
New List: 81 78
New List:

Final List:


Game over. No more pairs can be removed.
```

Lessons Learned:

In the code I worked on, I used a **ListIterator<Integer>** to navigate through a list of randomly generated integers, which simulated a solitaire matching game. This iterator allowed me to move forward and backward within the list, crucial for checking consecutive pairs of numbers to determine if they could be removed according to the game's rules.

Additionally, I implemented a new list strategy where numbers that couldn't be removed were stored, and this new list was displayed only when pairs were successfully removed. By using **iterator.next()** and **iterator.previous()** appropriately, I could efficiently modify the list as pairs were removed according to the game's rules. This iterative approach not only demonstrated the flexibility of iterators in dynamic list manipulation but also sharpened my ability to implement complex game logic in Java.