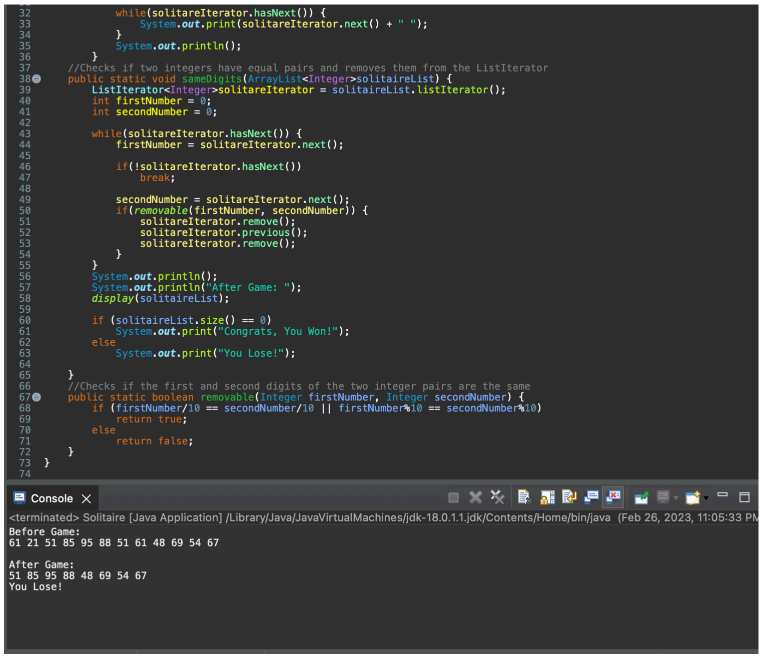
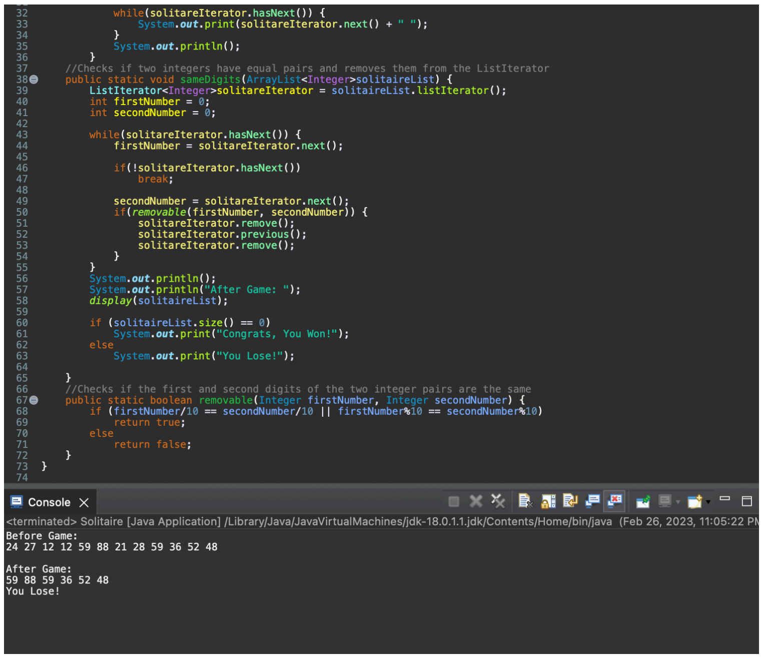
Write Up:
I haven't heard of ListIterators or Iterators in general before this week's lab. I learned that iterators are objects that traverse a collection of data. During iteration, each data item is considered once. I learned that iterators can traverse the elements in a collection only in the forward direction while ListIterator is able to traverse both forward and backwards.The add method inserts an entry into the list just before the iterator's current position. The remove method removes the list entry that last calls to either the next or previous returned. The set method replaces the entry that either is next or previous just returned. You can implement an iterator class inside the data structure class, as an inner class. This helps maintain the data structure's encapsulation. This is an integral iterator. You can also have it out the data structure class. This allows the client to specify the name of the iterator objects, can be used by multiple data structures and the implementation un-encapsulates the data structure. For this lab, it was a nice refresher on creating random integers. I had to remember the formula for using Random and setting it to (Maximum-Minimum)+Minimum. For the removable method, at first, I parsed the integers into strings and used the substrings to assign each digit. This was a pretty ugly way of doing it so I thought for awhile of another way without changing them to strings. I realized I could use division to check the first digit between the two pairs. I could then use modulo to check the second digit between the two pairs as well. This shrunk the code in half so I am definitely going to be remembering this little work around. The display method was pretty straight forward. Create the ListIterator based off the given ArrayList. Used a while statement to check if there is a next element and then to print that element. For the sameDigits method, everything was going smoothly until I kept getting the IllegalStateException. In my if statement, I had two ListIterator.remove() methods back to back. I figured this was the correct way to remove the two integers in the list, but kept getting an error message. I first put a next() method between the two but kept getting NoSuchElement error message. I then tried the previous() method between the two which then worked. I still little confused as to why the cursor needs to point back to the previous. In my theory, the first remove() method removes the second integer. Then the previous method is called to point to our first integers in which we then use the remove() method again. That's the only logical thing I can think of right now. I will attend office hours to clarify with you. I ran the simulations only using two numbers to test if my "Congrats, You Won" message worked which it did. Out of all my attempts with the 12 integers, I didn't win a single game.



```
		while(solitareIterator.hasNext()) {
			System.out.print(solitareIterator.next() + " ");
		}
		System.out.println();
	}
	//Checks if two integers have equal pairs and removes them from the ListIterator
	public static void sameDigits(ArrayList<Integer>solitaireList) {
		ListIterator<Integer>solitareIterator = solitaireList.listIterator();
		int firstNumber = 0;
		int secondNumber = 0;

		while(solitareIterator.hasNext()) {
			firstNumber = solitareIterator.next();

			if(!solitareIterator.hasNext())
				break;

			secondNumber = solitareIterator.next();
			if(removable(firstNumber, secondNumber)) {
				solitareIterator.remove();
				solitareIterator.previous();
				solitareIterator.remove();
			}
		}
		System.out.println();
		System.out.println("After Game: ");
		display(solitaireList);

		if (solitaireList.size() == 0)
			System.out.print("Congrats, You Won!");
		else
			System.out.print("You Lose!");

	}
	//Checks if the first and second digits of the two integer pairs are the same
	public static boolean removable(Integer firstNumber, Integer secondNumber) {
		if (firstNumber/10 == secondNumber/10 || firstNumber%10 == secondNumber%10)
			return true;
		else
			return false;
	}
}
```

Console

```
<terminated> Solitaire [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home/bin/java  (Feb 26, 2023, 11:05:33 PM
Before Game:
61 21 51 85 95 88 51 61 48 69 54 67

After Game:
51 85 95 88 48 69 54 67
You Lose!
```



```
		while(solitareIterator.hasNext()) {
			System.out.print(solitareIterator.next() + " ");
		}
		System.out.println();
	}
	//Checks if two integers have equal pairs and removes them from the ListIterator
	public static void sameDigits(ArrayList<Integer>solitaireList) {
		ListIterator<Integer>solitareIterator = solitaireList.listIterator();
		int firstNumber = 0;
		int secondNumber = 0;

		while(solitareIterator.hasNext()) {
			firstNumber = solitareIterator.next();

			if(!solitareIterator.hasNext())
				break;

			secondNumber = solitareIterator.next();
			if(removable(firstNumber, secondNumber)) {
				solitareIterator.remove();
				solitareIterator.previous();
				solitareIterator.remove();
			}
		}
		System.out.println();
		System.out.println("After Game: ");
		display(solitaireList);

		if (solitaireList.size() == 0)
			System.out.print("Congrats, You Won!");
		else
			System.out.print("You Lose!");

	}
	//Checks if the first and second digits of the two integer pairs are the same
	public static boolean removable(Integer firstNumber, Integer secondNumber) {
		if (firstNumber/10 == secondNumber/10 || firstNumber%10 == secondNumber%10)
			return true;
		else
			return false;
	}
}
```

Console

```
<terminated> Solitaire [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home/bin/java  (Feb 26, 2023, 11:05:22 PM
Before Game:
24 27 12 12 59 88 21 28 59 36 52 48

After Game:
59 88 59 36 52 48
You Lose!
```