Singleton Design Pattern Example

Step 1: create Singleton Class

```
public class Singleton {
    private static Singleton onlyInstance;
    private ArrayList<String >usernameList;
    private ArrayList<String> passwordList;
    private ArrayList<String> nameList;
    // MORE CODE
}
```

Step 2: create instance

```
class Singleton {
        …
        …
      private Singleton(){
            usernameList = new ArrayList<String>();
            passwordList = new ArrayList<String>();
            nameList = new ArrayList<String>();
      }

      public static Singleton getOnlyInstance() {
            if(onlyInstance == null)
                  onlyInstance = new Singleton();
            return onlyInstance;
      }
        …
        …
}
```

Step 3: getter, setter, updaters, …

```
class Singleton {
        …
        …
          public static void setOnlyInstance(Singleton onlyInstance) {
                SingletonLogin.onlyInstance = onlyInstance;
        }

        public  void addUsername(String username) {
                usernameList.add(username);
        }

        public void setUsename(int index, String newUsername){
                usernameList.set(index, newUsername);
        }

        public void removeUsename(int index, String newUsername){

        }


}
```

Sep 4: using Singleton Patten

```
public class DemoSingletoPattern {

    public static void main(String[] args) {

        Singleton c1 = Singleton.getOnlyInstance();
        c1.addName("Jack Lemon");
        c1.addUsername("jlemon");
        c1.addPassword("jl1234");

        Singleton c2 = Singleton.getOnlyInstance();
        c2.addName("Merry Leu");
        c2.addUsername(mleu);
         c2.addPassword("orange1234);

    }
}
```