CSCI 3901
Lab 1- Report

Akshat Gulati - B00863868

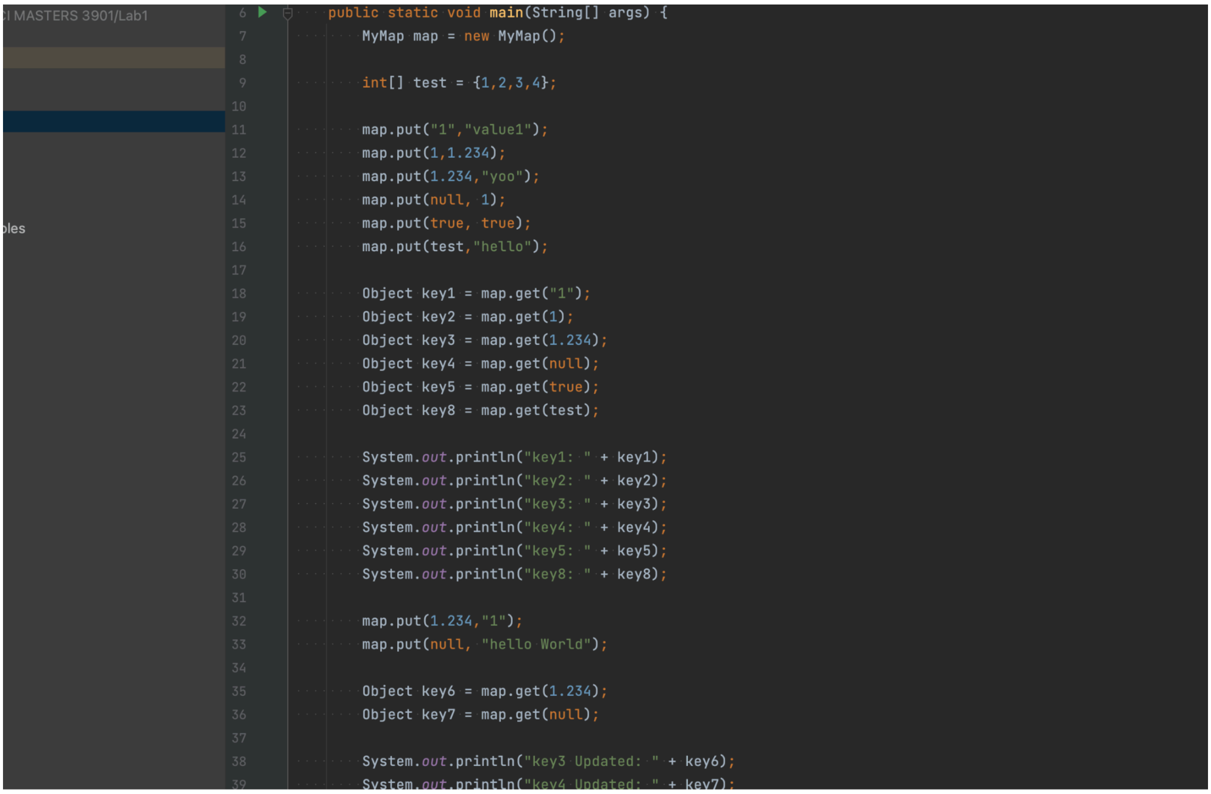
1. **Team Members:**
   - I did not have a Team Member but I discussed the solution in a group of 3 one day before without seeing any code and only focused on developing the logic by discussing the objective and understanding the output that the map should produce.
   - I developed the solution / coded the solution in the Lab itself.
   - **Study Group Members:**
     - Ritvik Wuyyuru - B01021909
     - Krishna Tej Nanda Kumar - B00975537

2. **Items from the Description that Needed Clarification:**
   - I asked the TA if we are allowed to implement the map as a dictionary in Python for Java allowing multiple key-value pairs of different primitive data types.
   - I asked if we are allowed to use the Object data type to implement my Custom Map.

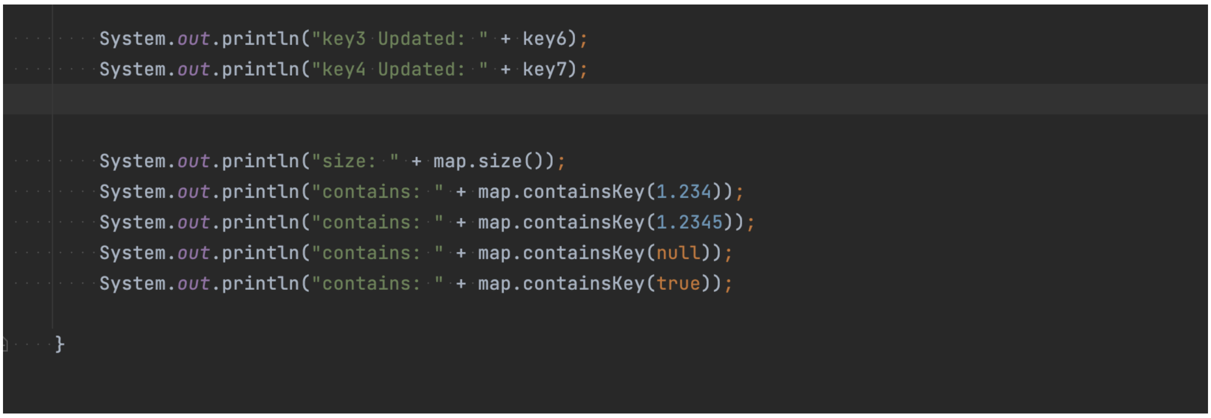
3. **Decisions on the Items That Needed Clarification:**
   - **Map as Dictionary:** After discussing with the TA, we decided to make it as a dictionary accepting different data types as part of a single Map object.
   - **Object Data Type:** Since it is not a part of Map or Set Classes we are allowed to use it.

4. **How We Showed That Our Work is Functional (So Far):**
   - Ran smoke tests to ensure basic functionality: get, put, size, and containsKey, and identify any immediate issues with the implementation.
   - Executed general test cases using common key-value pairs like <Integer, String>, <String, String>, <String, Integer>, and <Integer, Integer>, to verify reliable behavior across standard data types.
   - Tested the containsKey method with Float/Double keys to ensure it returned the correct boolean value and matched the expected output.
   - Tested the put method to confirm proper insertion and updating. When a key existed, the value was updated, and when the key was absent, a new key-value pair was inserted.
   - Tested with Float/Double as keys to assess how the implementation handled floating-point precision issues.

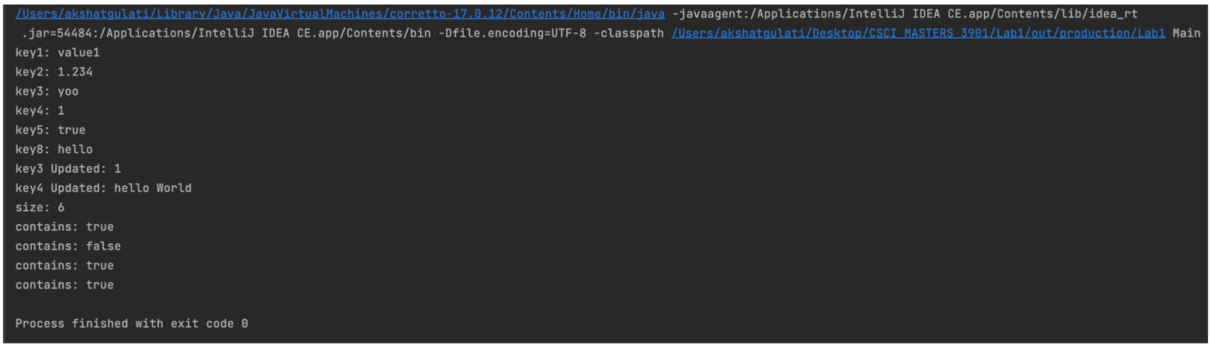
- Tested with arrays as keys to verify the implementation’s compatibility with non-primitive data types.



```java
    public static void main(String[] args) {
        MyMap map = new MyMap();

        int[] test = {1,2,3,4};

        map.put("1","value1");
        map.put(1,1.234);
        map.put(1.234,"yoo");
        map.put(null, 1);
        map.put(true, true);
        map.put(test,"hello");

        Object key1 = map.get("1");
        Object key2 = map.get(1);
        Object key3 = map.get(1.234);
        Object key4 = map.get(null);
        Object key5 = map.get(true);
        Object key8 = map.get(test);

        System.out.println("key1: " + key1);
        System.out.println("key2: " + key2);
        System.out.println("key3: " + key3);
        System.out.println("key4: " + key4);
        System.out.println("key5: " + key5);
        System.out.println("key8: " + key8);

        map.put(1.234,"1");
        map.put(null, "hello World");

        Object key6 = map.get(1.234);
        Object key7 = map.get(null);

        System.out.println("key3 Updated: " + key6);
        System.out.println("key4 Updated: " + key7);
```



```java
        System.out.println("key3 Updated: " + key6);
        System.out.println("key4 Updated: " + key7);


        System.out.println("size: " + map.size());
        System.out.println("contains: " + map.containsKey(1.234));
        System.out.println("contains: " + map.containsKey(1.2345));
        System.out.println("contains: " + map.containsKey(null));
        System.out.println("contains: " + map.containsKey(true));

    }
```



```
/Users/akshatgulati/Library/Java/JavaVirtualMachines/corretto-17.0.12/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt
 .jar=54484:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/akshatgulati/Desktop/CSCI MASTERS 3901/Lab1/out/production/Lab1 Main
key1: value1
key2: 1.234
key3: yoo
key4: 1
key5: true
key8: hello
key3 Updated: 1
key4 Updated: hello World
size: 6
contains: true
contains: false
contains: true
contains: true

Process finished with exit code 0
```

Analysis

5. **Identify how you will know that your implementation is working.**
   To determine if the implementation is working, I will:

   - **Run Smoke Tests**: Verify basic functionality by testing operations such as get, put, size, and containsKey, and identify any immediate issues.
   - **Execute General Test Cases**: Test with common key-value pairs like <Integer, String>, <String, String>, <String, Integer>, and <Integer, Integer> to ensure reliable behavior across standard data types.
   - **Test containsKey with Float/Double**: Check that this method returns the correct boolean value for Float/Double keys and matches the expected results.
   - **Test put Method**: Confirm proper functionality for inserting and updating values, ensuring that existing keys are updated and new keys are added as expected.
   - **Test with Float/Double as Keys**: Assess how the implementation handles floating-point precision issues.
   - **Test with Arrays as Keys**: Verify compatibility with non-primitive data types by using arrays as keys.

6. **Assurance of Code Quality:**
   - The code follows important design principles like keeping each part focused on one job (Single Job/Responsibility) and making it easy to add new features without changing existing code (OCP).
   - I tested the code thoroughly with basic tests, edge cases, and using different key types like Float/Double and arrays. This shows the map works well in various situations.
   - The class is easy to understand, with clear names for methods and straightforward logic, making it easy to read and maintain.
   - I used two classes, KeyValue and MyMap, to keep things organized, following a "Noun-Verb" approach (things and actions).
   - I kept the classes in separate files to make the project more readable and maintainable (Modularization).
   - Using a LinkedList makes sure that adding, and going through items happens efficiently, which is important for how the map works.
   - I also tested the LinkedList for performance, especially to check that key operations like containsKey and put work quickly for common use cases.

7. **Difficulties Encountered and How We Dealt with Them:**

- Choosing a design approach to keep the code straightforward and functional.
- Selecting a data structure to replace the map, like ArrayList, LinkedList, 2D array, or parallel arrays.
- Managing different data types, deciding between method overloading, templating, or using Object for the map.
- Handling null keys effectively.
- Addressing floating-point precision issues with Float/Double.
- Working with BigDecimal values.
- Utilizing the Objects class.
- Identifying possible edge cases.
- Comparing BigDecimal values correctly.

8. Reflection:

What We Did Well:

I’ve figured out how to use the `Objects` class and handle floating-point precision issues by using BigDecimal instead, which is useful for verifying conditions in coding problems without rounding or approximation. The `Objects` class helps create generalized solutions when the data type is unknown and enables performing tasks within a single function rather than relying on method overloading.