Project 1 - part 2

Name: Vishal Prabhachandar
Section: CSCI 5448-001
JDK version: Open JDK Amazon Corretto-11.0.13.8.1

1. Playing card

The first 3 pictures contain the code for the problem.
Followed by 4 pictures that contain the output from the program for the defined inputs.
Kindly scroll below to see the screenshots.

PlayingCards.java

```java
/*
Author: Vishal Prabhachandar
Section: CSCI 5448-001
JDK version: Open JDK Amazon Corretto-11.0.13.8.1
Playing cards program

Assumptions:
1. The program ends when the user inputs 0.
2. Deck is reinitialized after selection
        - Example: when 5 random cards are shown from the deck. Before next random selection based on input
                   the deck is reset.
3. Arraylist is used to store the deck of cards.
4. Used Static variables and methods to hold card deck, so only single object is shared among all objects.
5. No code was copied, looked up documentation to perform operations.
*/

import java.util.ArrayList;
import java.util.Collections;
import java.util.Random;
import java.util.Scanner;

// Class to hold card details
class Deck{
    static String[] SUIT = {
            "Club", "Diamond", "Heart", "Spade"
    };

    static String[] FACE_VALUE = {
            "A", "2", "3", "4", "5", "6", "7", "8", "9", "10",
            "J", "Q", "K"
    };
    static int DECK_SIZE = 54;
    // Used to store the cards
    static ArrayList<String> DECK = new ArrayList<>();
    // Function to initialize the DECK of cards
    static void initialize(){
        DECK.clear();
        for (String s : FACE_VALUE) {
            for (String value : SUIT) {
                DECK.add(s + "-" + value);
```

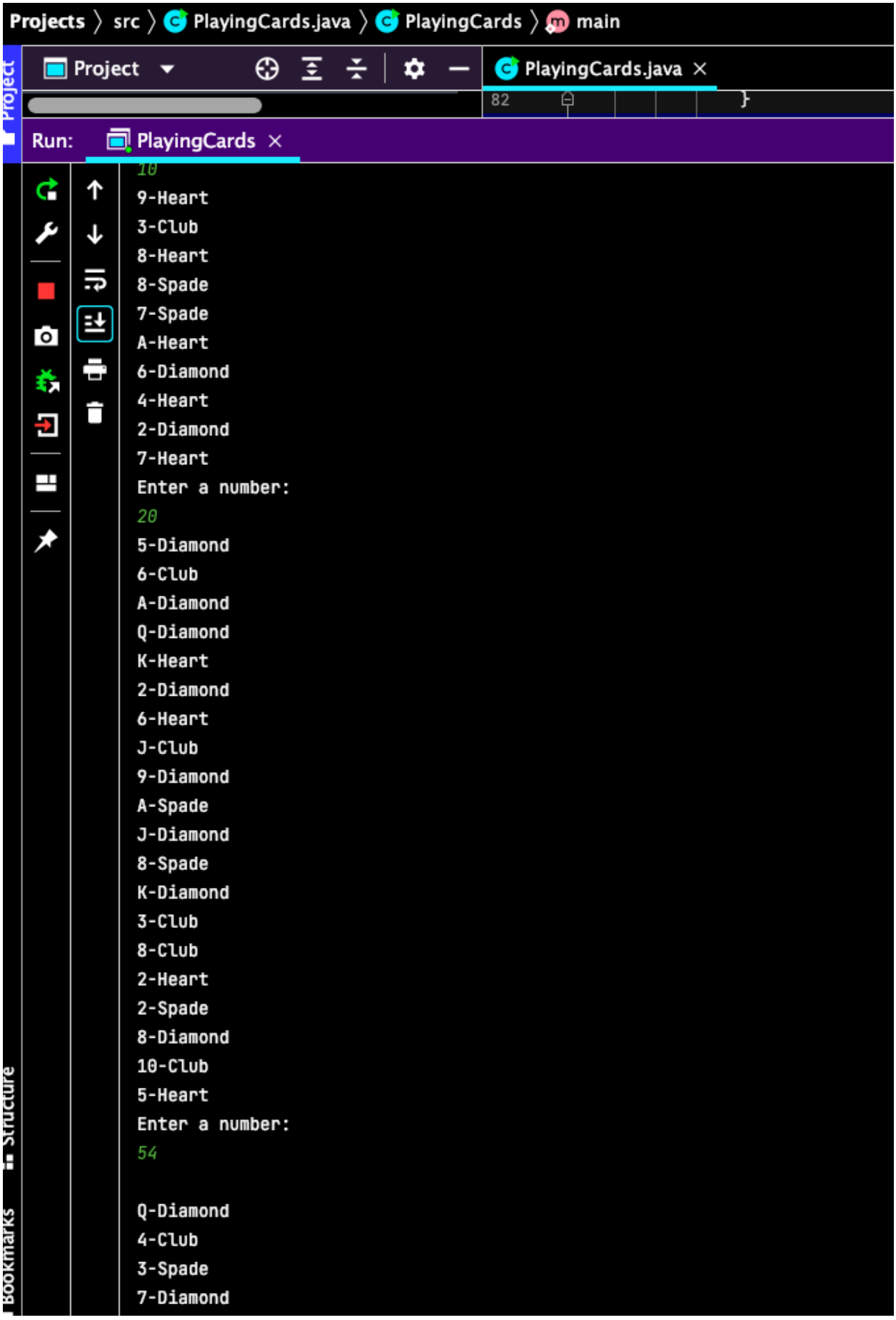
PlayingCards.java

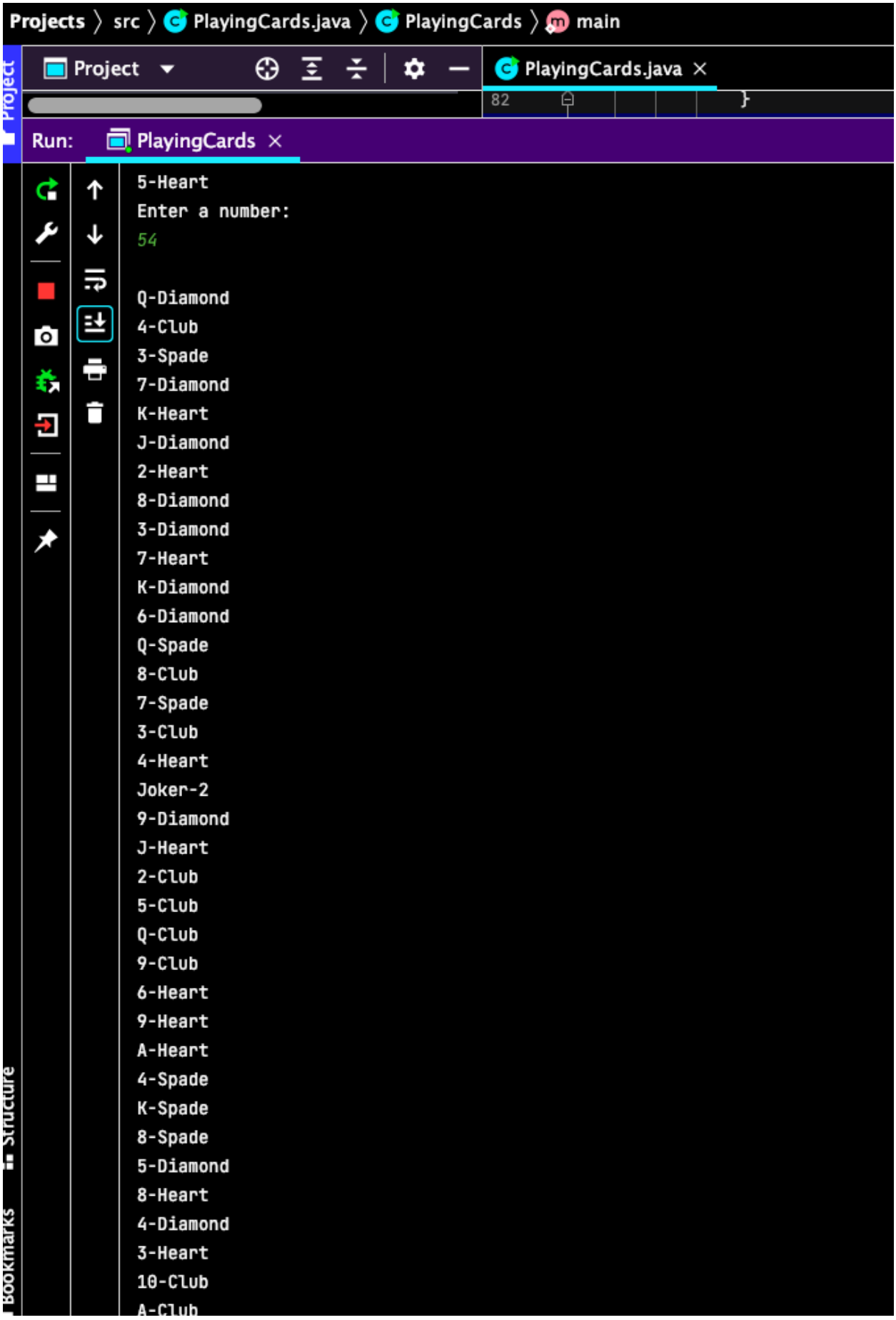
```java
    static void initialize(){
        DECK.clear();
        for (String s : FACE_VALUE) {
            for (String value : SUIT) {
                DECK.add(s + "-" + value);
            }
        }
        // Adding the two jokers
        DECK.add("Joker-1");
        DECK.add("Joker-2");
    }
    // Function to shuffle the DECK for more randomness
    static void shuffleDeck(){
        Collections.shuffle(DECK);
    }
}
// Class to hold methods to perform random selection from DECK
class randomCardSelector{
    void printRandomCard(int number){
        // Random generator
        Random randomIndex = new Random();
        int deckIndex;
        for (int cardNumber = 0; cardNumber < number; cardNumber++) {
            // find random index in arraylist to print
            deckIndex = randomIndex.nextInt( bound: Deck.DECK_SIZE - cardNumber);
            System.out.println(Deck.DECK.get(deckIndex));
            // Remove the shown card from arraylist so there is no repetition
            Deck.DECK.remove(deckIndex);
        }
    }
}
// Driver class
public class PlayingCards {
    public static void main(String[] args) {
        // Initializing deck - there is no need to create an object to initialize as all the values are static
        Deck.initialize();
        // Shuffling deck
        Deck.shuffleDeck();
        // Object creation for randomCardSelector
        randomCardSelector operation = new randomCardSelector();
```

PlayingCards.java

```java
public class PlayingCards {
    public static void main(String[] args) {
        // Initializing deck - there is no need to create an object to initialize as all the values are static
        Deck.initialize();
        // Shuffling deck
        Deck.shuffleDeck();
        // Object creation for randomCardSelector
        randomCardSelector operation = new randomCardSelector();
        int inputNumber = -1;
        Scanner input = new Scanner(System.in);
        // Accepting input until 0
        do{
            System.out.println("Enter a number: ");
            // Try catch for handling inputs other than numbers
            try{
                inputNumber = input.nextInt();
                // Checking for valid input range
                if (inputNumber < 0 || inputNumber > 54){
                    System.out.println("Enter a valid integer between 0 and 54!");
                }
                // Check if number is not 0 and perform random selection
                else if (inputNumber != 0 ){
                    operation.printRandomCard(inputNumber);
                    Deck.initialize();
                }
            }
            catch (Exception e){
                System.out.println("Enter a valid integer between 0 and 54!");
                input.next();
            }
        }while(inputNumber != 0);

    }
}
```


Projects › src › PlayingCards.java › PlayingCards › main
Project
PlayingCards.java
Run: PlayingCards
/Library/Java/JavaVirtualMachines/amazon-corretto-11.jdk/Contents/Home/bin
Enter a number:
-3
Enter a valid integer between 0 and 54!
Enter a number:
1
4-Heart
Enter a number:
5
Joker-2
2-Spade
Joker-1
3-Club
6-Spade
Enter a number:
10
9-Heart
3-Club
8-Heart
8-Spade
7-Spade
A-Heart
6-Diamond
4-Heart
2-Diamond
7-Heart
Enter a number:
20
5-Diamond
6-Club
A-Diamond
Q-Diamond
K-Heart
2-Diamond
6-Heart
J-Club
9-Diamond
A-Spade
J-Diamond
8-Spade
Structure
Bookmarks


Projects
src
PlayingCards.java
PlayingCards
main
Project
PlayingCards.java
Run:
PlayingCards
10
9-Heart
3-Club
8-Heart
8-Spade
7-Spade
A-Heart
6-Diamond
4-Heart
2-Diamond
7-Heart
Enter a number:
20
5-Diamond
6-Club
A-Diamond
Q-Diamond
K-Heart
2-Diamond
6-Heart
J-Club
9-Diamond
A-Spade
J-Diamond
8-Spade
K-Diamond
3-Club
8-Club
2-Heart
2-Spade
8-Diamond
10-Club
5-Heart
Enter a number:
54
Q-Diamond
4-Club
3-Spade
7-Diamond
Structure
Bookmarks


Projects › src › PlayingCards.java › PlayingCards › main
Project
PlayingCards.java
82
}
Run:
PlayingCards
5-Heart
Enter a number:
54
Q-Diamond
4-Club
3-Spade
7-Diamond
K-Heart
J-Diamond
2-Heart
8-Diamond
3-Diamond
7-Heart
K-Diamond
6-Diamond
Q-Spade
8-Club
7-Spade
3-Club
4-Heart
Joker-2
9-Diamond
J-Heart
2-Club
5-Club
Q-Club
9-Club
6-Heart
9-Heart
A-Heart
4-Spade
K-Spade
8-Spade
5-Diamond
8-Heart
4-Diamond
3-Heart
10-Club
A-Club
Project
Structure
Bookmarks


Projects › src › PlayingCards.java › PlayingCards › main
Project
PlayingCards.java
82
}
Run:
PlayingCards
Q-Club
9-Club
6-Heart
9-Heart
A-Heart
4-Spade
K-Spade
8-Spade
5-Diamond
8-Heart
4-Diamond
3-Heart
10-Club
A-Club
Joker-1
5-Heart
J-Club
5-Spade
7-Club
A-Spade
2-Spade
A-Diamond
10-Diamond
10-Spade
6-Spade
2-Diamond
10-Heart
J-Spade
9-Spade
6-Club
Q-Heart
K-Club
Enter a number:
55
Enter a valid integer between 0 and 54!
Enter a number:
0
Process finished with exit code 0
Structure
Bookmarks

2. Input string matching

The first 4 pictures contain the code for the problem.
Followed by 2 pictures that contain the output from the program for the defined inputs.
Kindly scroll below to see the screenshots.

StringMatching.java

```java
/*

Author: Vishal Prabhachandar
Section: CSCI 5448-001
JDK version: Open JDK Amazon Corretto-11.0.13.8.1
String matching program

Assumptions:
1. The program ends when the user inputs a null/empty string or the input word matches the randomly selected word.
2. Static arrays is used to store the predefined 12 five-letter words.
3. All the predefined words are assumed to be in upper case and have non-repeating characters.
4. Only the input string is upper-cased.
*/

import java.util.HashSet;
import java.util.Random;
import java.util.Scanner;

// Class to hold list of predefined words
class WordList{
    // Storing the 12 words
    static String[] WORD_SET = { "BRUCE", "DWIGH", "MAXDO", "ROSHN", "GAPBN", "FRANK",
            "PHONE", "WORDS", "CREAM", "PASTE", "DOUGH", "DOGIE"};

    String getWord(){
        // Random generator
        Random randomIndex = new Random();
        int cardIndex = randomIndex.nextInt(WORD_SET.length);
        return WORD_SET[cardIndex];
    }
}

/* Class to perform the string comparison
    1. Hash set is used to store the characters of the randomly selected word (the random word doesn't contain repeating characters)
    2. Both input string and randomly selected word are converted to character array, so they are iterable
    3. We first check if there is an exact match of characters
    4. Otherwise, we look up the hashset using contains function to find if a character is present
    5. Finally, we conclude the character is not present
```

StringMatching.java

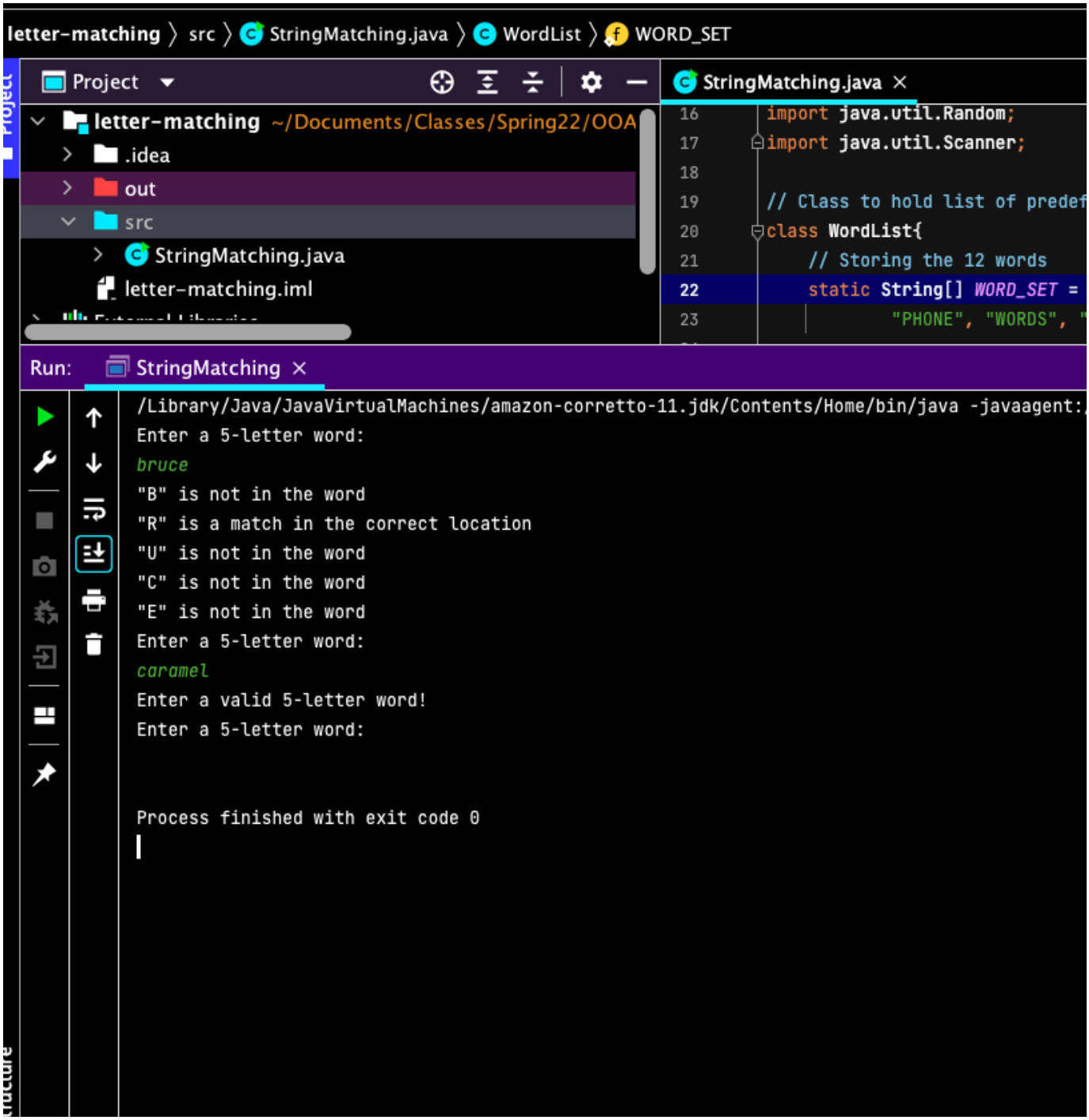
```
2. Both input string and randomly selected word are converted to character array, so they are iterable
3. We first check if there is an exact match of characters
4. Otherwise, we look up the hashset using contains function to find if a character is present
5. Finally, we conclude the character is not present

*/

class CompareStrings {

    boolean run(String randomWord, String inputString) {

        HashSet<Character> charLocation = new HashSet<>();
        int countMatch = 0;
        char[] randomWordChars = randomWord.toCharArray();
        char[] inputStringChars = inputString.toUpperCase().toCharArray();
        for (char c : randomWordChars) {
            charLocation.add(c);
        }
        for (int index = 0; index < inputString.length(); index++) {
            if (randomWordChars[index] == inputStringChars[index]) {
                ++countMatch;
                System.out.println("\"" + inputStringChars[index] + "\"" + " is a match in the correct location");
            } else if (charLocation.contains(inputStringChars[index])) {
                System.out.println("\"" + inputStringChars[index] + "\"" + " is in the word but in a different location");
            } else {
                System.out.println("\"" + inputStringChars[index] + "\"" + " is not in the word");
            }
        }
        // Code to return boolean value, if all characters match inorder to exit the program
        return countMatch == inputString.length();
    }
}
// Class for custom exception
// Reference - https://stackoverflow.com/questions/8423700/how-to-create-a-custom-exception-type-in-java
class wordException extends Exception{

    public wordException(String message){
        super(message);
```

StringMatching.java

```java
class wordException extends Exception{

    public wordException(String message){
        super(message);
    }
}

// Driver class
public class StringMatching {

    public static void main(String[] args){

        // getting random word
        WordList words = new WordList();
        String randomWord;
        // Object for CompareStrings class
        CompareStrings compare = new CompareStrings();
        // Flag variable to denote end of program or complete word match
        boolean isEnd;
        String inputString;
        Scanner input = new Scanner(System.in);
        // Accepting input until empty or word matches random word
        do{
            // Setting isEnd flag to false
            isEnd = false;
            // Set random word
            randomWord = words.getWord();
            System.out.println("Enter a 5-letter word: ");
            // Try catch for handling inputs other than numbers

            inputString = input.nextLine();
            // Checking for empty input
            if (inputString.isBlank() || inputString.isEmpty()){
                isEnd = true;
            }
            // Check if words are alphabets only
            // Reference - https://www.tutorialkart.com/java/how-to-check-if-string-contains-only-alphabets-in-java/
            else if(!inputString.matches(regex: "[a-zA-Z]+")){
```

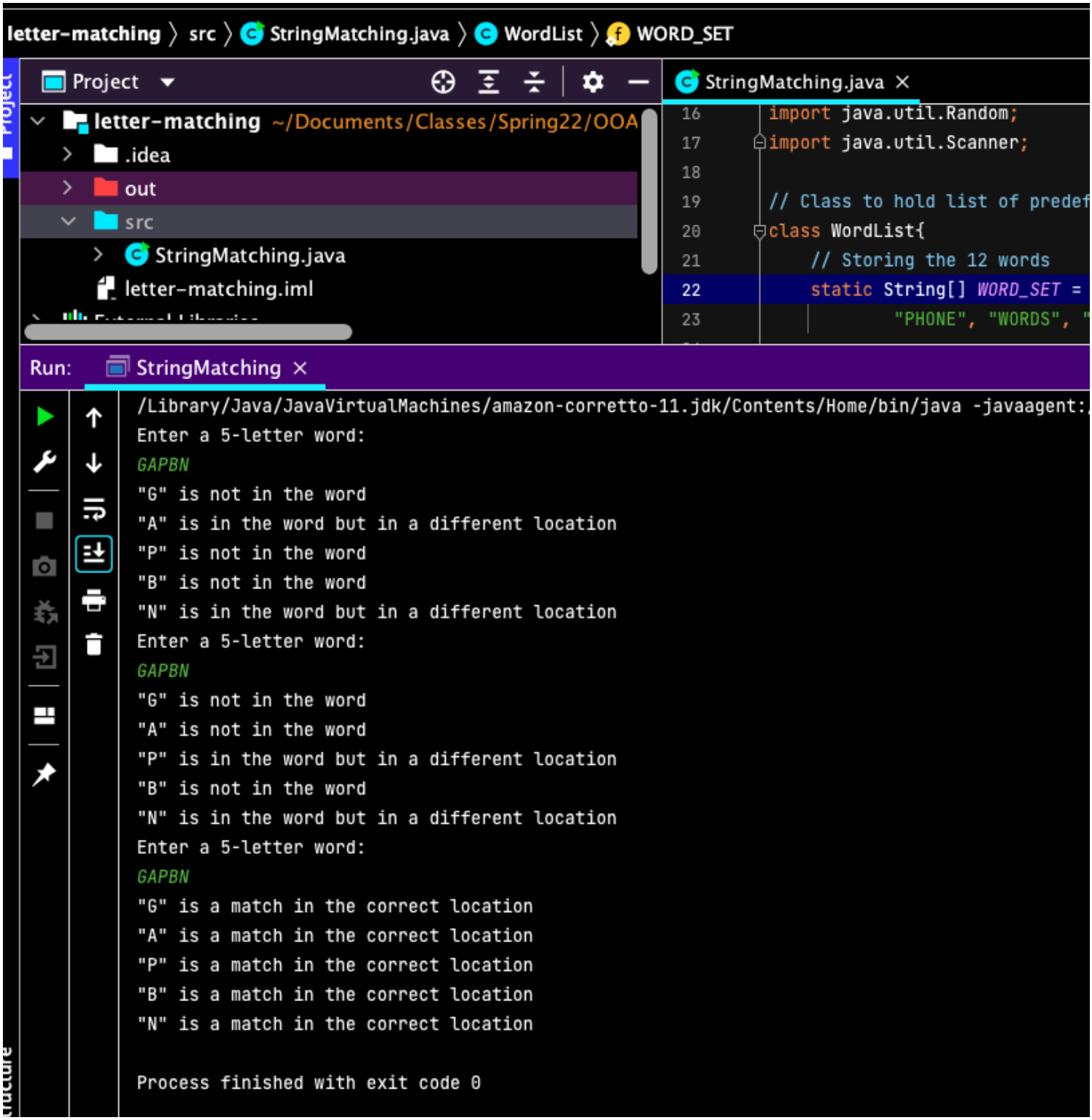
StringMatching.java

```
            // Set random word
            randomWord = words.getWord();
            System.out.println("Enter a 5-letter word: ");
            // Try catch for handling inputs other than numbers

            inputString = input.nextLine();
            // Checking for empty input
            if (inputString.isBlank() || inputString.isEmpty()){
                isEnd = true;
            }
            // Check if words are alphabets only
            // Reference - https://www.tutorialkart.com/java/how-to-check-if-string-contains-only-alphabets-in-java/
            else if(!inputString.matches( regex: "[a-zA-Z]+")){
                try{
                    throw new wordException("Enter a valid 5-letter word!");
                }
                catch(wordException we){
                    System.out.println(we.getMessage());
                }
            }
            // Checking for valid input length
            else if (inputString.length() != 5){
                try{
                    throw new wordException("Enter a valid 5-letter word!");
                }
                catch(wordException we){
                    System.out.println(we.getMessage());
                }
            }
            // Perform comparison
            else{
                isEnd = compare.run(randomWord, inputString);
            }

        }while(!isEnd);
    }
}

```



The above screenshot is the output for

1. Match in the correct location
2. Not in word
3. Wrong size word
4. Exit with null word input



The above screenshot is the output for

5. Word in a different location
6. Matching the entire word