**PALINDROME LAB INSTRUCTIONS**

This question involves identifying palindromes. A *palindrome* is a string of characters that reads the same from front-to-back and back-to-front. Three examples of palindromes are **racecar**, **madam** and **123454321**.

These first three examples are perfect palindromes. It is possible to read the string of characters from either end with the exact same results, without any special conditions. There are other strings of characters, which can be considered palindromes, but they are not perfect. Three examples of imperfect palindromes are **Racecar**, **Madam I'm Adam** and **A man, a plan, a canal, Panama**. **Racecar** can be considered a palindrome if case-sensitivity is ignored. The other two examples are also palindromes if case-sensitivity, spaces and punctuation are ignored.

For this question a palindrome shall be defined as a *string of alpha-numeric characters that reads the same from front-to-back and back-to-front without case-sensitivity*. This definition means that strings like **Racecar** and **Madam** will be considered palindromes, but **Madam I'm Adam** is not a palindrome.

Consider the following incomplete declaration of a Palindrome class. The constructor determines if the parameter is a palindrome by using the isPalindrome method. The private method isPalindrome needs to return true if the string is a palindrome and false otherwise. The method toString needs to be redefined to produce the output shown in the execution example.

class Palindrome

{

private boolean palindrome;

private String str;

Palindrome(String s)

{

str = s;

palindrome = isPalindrome();

}

public String toString()

{

**/\* to be redefined in part (a) \*/**

}

private boolean isPalindrome()

{

**/\* to be implemented in part (b) \*/**

}

}

The code segment below is part of a client program that constructs and displays two Palindrome objects.

|  |  |
| --- | --- |
| **Code segment in client program using the Palindrome class** | **Execution Output** |
| Palindrome p1 = new Palindrome("Racecar");  System.out.println(p1);  Palindrome p2 = new Palindrome("Madam I'm Adam");  System.out.println(p2); | String: Racecar  Palindrome: true  String: Madam I'm Adam  Palindrome: false |

Part (a).

Redefine the toString method as discussed earlier. Method toString must return a string that will display two lines of output. The following table shows the result of several different calls made by System.out.println(/\* Palindrome Object \*/) based on the argument of the constructor.

|  |  |
| --- | --- |
| **Constructor argument** | **toString returns** |
| Racecar | String: Racecar  Palindrome: true |
| 12345BOB54321 | String: 12345BOB54321  Palindrome: true |
| Madam I'm Adam | String: Madam I'm Adam  Palindrome: false |
| MADAM RACECAR | String: MADAM RACECAR  Palindrome: false |

Complete method toString below.

**/\*\***

**\* precondition:** str is not null

\* palindrome is true or false

\* **Postcondition:** returns a string in the format:

\* String: value of str

\* Palindrome: value of palindrome

\*/

public String toString()

{

}

Part (b).

Write method isPalindrome as described earlier. In writing method isPalindrome you may use method toUpperCase which returns a string with all lower-case characters converter to upper-case characters.

EX: String a = “Hello”;

System.out.println(a.toUpperCase()); 🡪 HELLO

Complete method isPalindrome below.

/\*\*

\* **Precondition**: str is not null

\* **Postcondition**: returns true if str is a palindrome, and false otherwise

\*/

private boolean isPalindrome()

{

}