COMP2026 Problem Solving Using Object Oriented Programming

Laboratory 6

Part A Discovery Exercises

Task 1: Methods

a) Given:

```
boolean eq3(int a, int b, int c) {
    if(a == b) && (a == c)){
        return true;
    }
    return false;
}
```

What does the above method do?

b) Given:

```
...
void runApp(){
    int x = 5;
    System.out.println(func(x));
    System.out.println(func(x+3));
    System.out.println(func(func(x+3)));
}
int func(int a) {
    return a + 3;
}
...
```

What values are printed out by the above code fragment?

c) Given:

```
...
void runApp(){
    int a = 1;
    int b = 2;

    System.out.println("Before update...");
    System.out.println("a = " + a);
    System.out.println("b = " + b);

    update(a, b);

    System.out.println("After update...");
    System.out.println("a = " + a);
    System.out.println("b = " + b);
}

void update(int a, int b)
{
    a++;
    b++;
    System.out.println("In update...");
    System.out.println("a = " + a);
    System.out.println("b = " + b);
}
```

Write the output in the box below.

d) Given:

```
...
 void runApp(){
    int[] a = {1, 2};

    System.out.println("Before update...");
    printArray(a);

    update(a);

    System.out.println("After update...");
    printArray(a);
}

void update(int[] a)
{
    for(int i = 0; i < a.length; i++)
    {
        a[i]++;
    }

    System.out.println("In update...");
    printArray(a);
}


void printArray(int[] a)
{
    for(int i = 0; i < a.length; i++)
    {
        System.out.println("a[" + i + "] = " + a[i]);
    }
}
```

Write the output in the box below.

Task 2: Method Overloading

Given:

```
...
String fun(){
    return "Java";
}

String fun(String a, int b){
    return a + " " + b;

}

String fun(String s){
    String r = "";
    for(int i = 0; i < 5; i++){
        r = r + s;
    }
    return r;
}

String fun(int x){
    if(x > 80){
        return "High!";
    }
    return "Low!";
}

String fun(char c){
    String s = "";
    for(int i = 0; i < 3; i++){
        s = s + "-" + c;
    }
    return s;
}
...
```

What will be printed out by the following code fragment?

```
System.out.println(fun());
System.out.println(fun('T'));
System.out.println(fun("Java", 2026));
System.out.println(fun("Assignment"));
System.out.println(fun("A"));
System.out.println(fun(100));
```

Answer:

Part B Programming Exercises

Task 1: Writing Methods

Write the following methods in the given **`MyMethods.java`**.

a) Write a method call **`CelToFah`** to convert temperature in Celsius to Fahrenheit. The temperature in degrees Fahrenheit (°F) is equal to the temperature in degrees Celsius (°C) times 9/5 plus 32.
b) A palindrome is a sequence of characters that reads the same both forward and backward. Some examples are
"otto" "level" "refer" "121" "i may yam i"
Write an **`isPalindrome`** method that takes an array of **`char`** and returns the **`boolean`** value true if the sequence of characters is a palindrome.
c) Write a method call **`strToInt`** to convert a String with digits, such as "123" into an integer type, 123.
Note:
- You are **NOT allowed** to use the java built in methods to parse numerical values from Strings. E.g. parseInt(), parseDouble(), etc.
- You can use the String method **`length()`** and **`charAt()`**.

Sample output:

```
37.5 in degree Celsius equals 99.5 in degree Fahrenheit.
otto is palindrome.
level is palindrome.
java is not palindrome.
Sum of 12345 and  60798 is: 73143
```

Task 2: Maze Game

Maze is a complicated system of the path. The player wins if they move from the starting point to the destination. Write a program called **`Maze`** that reads in a r×c maze(r and c indicated at the first line) from a text file and stores the maze in a r×c character array. Then it prints the character array and accepts inputs from players. The program should print the maze in each turn and stop the game if the player arrives at the destination.

Implement the following methods in the program:

- **`void printMaze(char[][] maze, int r, int c){…}`** that prints the Maze and the current position of the player as '*'. The given **`r`** and **`c`** are the current position's row number and the column number.
- **`int[] findStartPosition(char[][] maze){…}`** that finds the row number and column number of the starting point(S) in the maze, then returns the row number and column number as an integer array. The row number should in index 0 and the column number should be in index 1 of the array.

- **`char getMove(){…}`** that prompts user to enter the move, reads the player input and returns the input.
- **`int[] findNewPosition(int r, int c, char m){…}`** that finds the new row number and new column number after the move **`m`** is made on the current position. Then returns the new row number and column number as an integer array. The new row number should be in index 0 and new column number should be in index 1 of the array.
- **`boolean isValidMove(char[][] maze, int r, int c){…}`** that checks whether the given row number **`r`** and column number **`c`** is a valid move. The
- player is not allowed to move into the wall(X) and the boundaries.
- **`boolean isWin(char[][] maze, int r, int c){…}`** that checks whether the player arrives at the destination. The method also prints a "You Win!" message if the player arrives at the destination.

Sample output:

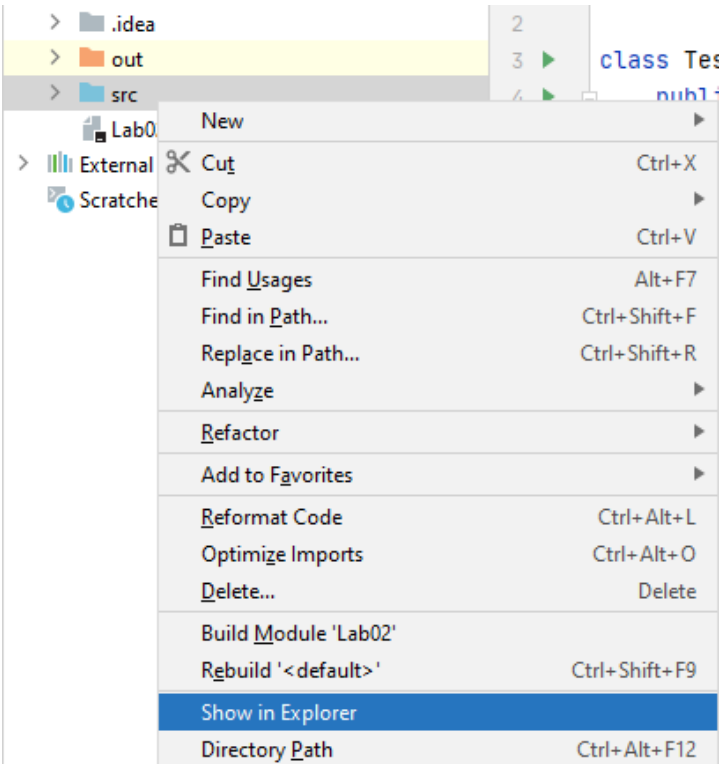
```
Enter the filename: Maze1.txt
*   X
      X
    X
      D
* is your current position.
Use a,s,d,w (a:left, s:down, d:right, w:up) to move to destination (D).
Enter your move (a,s,d,w): d

S * X
      X
    X
      D
Enter your move (a,s,d,w): d
Invalid Move!

S * X
      X
    X
      D
Enter your move (a,s,d,w): w
Invalid Move!

S * X
      X
    X
      D
```

```
Enter your move (a,s,d,w): s

S . X
  *   X
    X
      D
Enter your move (a,s,d,w): s

S . X
  .   X
  * X
      D
Enter your move (a,s,d,w): s

S . X
  .   X
  . X
  *   D
Enter your move (a,s,d,w): s
Invalid Move!

S . X
  .   X
  . X
  *   D
Enter your move (a,s,d,w): d

S . X
  .   X
  . X
  . * D
```

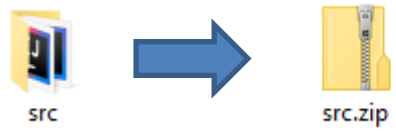
```
Enter your move (a,s,d,w): d

S . X
  .   X
  . X
  . . *
You Win!
```

Part C Submitting Exercises

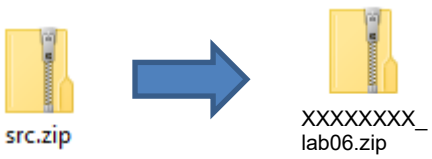
Step 1: Right-click the **src** folder and select **Show in Explorer**



Step 2: Zip the **src** folder into **src.zip**



Step 3: Rename the src.zip file to ***XXXXXXXX_lab06.zip*** where ***XXXXXXXX*** is your **student id**



Step 4: Submit ***XXXXXXXX_lab06*.zip** and **XXXXXXXX_lab06.docx** to Moodle.

References

[1] Bravaco, R., & Simonson, C. (2009). *Java programming: From the ground up*. Dubuque, IA: McGraw-Hill.

[2] Dean, J., & Dean, R. (2008). *Introduction to programming with Java: A problem solving approach*. Boston: McGraw-Hill.

[3] Farrell, J. (2012). *Java programming. Boston, MA: Course Technology Cengage Learning*

[4] Forouzan, B. A., & Gilberg, R. F. (2007). *Computer science: A structured programming approach using C (3rd ed.)*. Boston, MA: Thomson Course Technology.

[5] Gaddis, T. (2016). *Starting out with Java (6th ed.)*. Pearson.

[6] Liang, Y. D. (2013). *Introduction to Java programming: Comprehensive version*. (8th ed.). Pearson.

[7] Schildt, H. (2006). *Java a beginner's guide*. New York: McGraw Hill.

[8] Schildt, H., & Skrien, D. J. (2013). Java programming: A comprehensive introduction. New York: McGraw-Hill.

[9] Wu, C. T. (2010). *An introduction to object-oriented programming with Java*. Boston: McGraw Hill Higher Education

[10] Xavier, C. (2011). *Java programming: A practical approach*. New Delhi: Tata McGraw Hill.

[11] yet another insignificant Programming Notes. (n.d.). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming

[12] Zakhour, S., Kannan, S., & Gallardo, R. (2013). *The Java tutorial: A short course on the basics (5th ed.)*.