Practice Problems Week 3

Files, Strings, Exceptions

Josh Hernandez

COMP 1020
Introductory Computer Science 2

Predict Output - Section Overview

- The following questions present code snippets (sometimes with poor style).
- Your goal is to determine the exact output printed to the console.
- Pay attention to:
  - String immutability and the String Pool.
  - Exception propagation and control flow (try/catch/finally).
  - References vs Value comparison.

Problem 1: String Pools

```
String s1 = "Java";
String s2 = "Java";
String s3 = new String("Java");
String s4 = "Ja" + "va";

System.out.println((s1 == s2) + " " + s1.
    equals(s2));
System.out.println((s1 == s3) + " " + s1.
    equals(s3));
System.out.println((s1 == s4));
```

string objects

string pool

equals : compare the sequence of character

== : compare the reference

Predict the output.

true true
false true
true

Problem 2: Immutability

```
String s = "Hello";
s.concat(" World"); /concat(): nối chuỗi
s.toUpperCase(); // HELLO WORLD
String t = s.substring(0, 2);

System.out.println(s); → Hello
System.out.println(t); → Hel
```

String là immutable (không thể sửa trực tiếp)

concat() và toUpperCase() không thay đổi S

Nếu muốn đổi S ta phải gán lại một biến khác:

S_1 = S.concat(" world");

S_2 = S.toUpperCase();

Predict the output.

Problem 3: Exception Flow 1

```
try {
    System.out.print("A");
    String s = null;
    System.out.print(s.length());
    System.out.print("B");
} catch (ArithmeticException e) {
    System.out.print("C");
} catch (NullPointerException e) {
    System.out.print("D");
} finally {
    System.out.print("E");
}
System.out.print("F");
```

A D E F

Predict the output.

Problem 4: Exception Flow 2

```java
try {
    System.out.print("1");
    int x = 5 / 0;
    System.out.print("2");
} catch (RuntimeException e) {
    System.out.print("3");
} catch (ArithmeticException e) { // Compile Note
    System.out.print("4");
} finally {
    System.out.print("5");
}
```

1 4 5

Predict the output / behavior.

(Hint: Look closely at the catch block order)

Problem 5: String Methods

```java
String data = " A,B, C ";
String[] parts = data.trim().split(",");
System.out.print(parts.length + ":");
for(String p : parts) {
    System.out.print("[" + p + "]");
}
```

→ 3

→ [A]

[B]

[C]

space

Predict the output.

Problem 6: Exception Flow 3

```
public static void main(String[] args) {
    try {
        methodA();
    } catch (Exception e) {
        System.out.println("Main Catch");
    }
}
public static void methodA() {
    try {
        throw new RuntimeException();
    } finally {
        System.out.print("FinallyA ");
    }
}
```

output: Finally A Main Catch

Predict the output.

Problem 7: Scanner Tokens

```
// Input String: "10 20 Hello 30"
Scanner sc = new Scanner("10 20 Hello 30");
int sum = 0;
while (sc.hasNextInt()) {
    sum += sc.nextInt();
}
System.out.println(sum + " " + sc.next());
```

↳ vòng lặp dừng

→ đọc tới khi ko có số nguyên tiếp theo

0 += 10

10 += 20 → 30

Output : 30 Hello

Predict the output.

Problem 8: IndexOf and Substring

```
                0 1 2 3 4 5
String s = "banana";
int idx = s.indexOf('a'); → position of 1st 'a'
int last = s.lastIndexOf('a'); → position of the last 'a'
System.out.println(idx + " " + last);
System.out.println(s.substring(idx, last)); → anan
```

Predict the output.

Output: 1 5
anan

Problem 9: Pass by Value (Strings)

```
public static void main(String[] args) {
    String s = "Start";
    modify(s);
    System.out.println(s);
}
public static void modify(String str) {
    str = str + "End";
}
```

Output: Start -> 'S' is immutable

Predict the output.

Problem 10: Scanner Delimiter

```
// Input: "apple,banana,cherry"
Scanner sc = new Scanner("apple,banana,
    cherry");
sc.useDelimiter(",");
while(sc.hasNext()) {
    System.out.print(sc.next().charAt(0));
}                    Output: abc
```

Predict the output.

Fill In Code - Section Overview

- Complete the missing segments _______ to solve the problem.
- Follow strict Programming Standards (constants, spacing, variable names).

Problem 11: Safe Division

Task: Implement a method to divide two numbers, returning 0 on error.

```
public static int safeDivide(int top, int bottom) {
    int result = 0;

    // Fill in the try-catch block
    _____try___________ {
        result = top / bottom;
    } Catch (ArithmeticException e)___ {
        System.out.println("Cannot divide by zero.");
    }

    return result;
}
```

Problem 12: File Setup

Task: Setup a Scanner to read from "data.txt". Handle the checked exception.

```
public static void processFile() {
    Scanner fileReader = null;

    try {
        // Initialize Scanner for file "data.txt"
        fileReader = new Scanner (new File ("data.txt"))___;

    } catch (File Not Found Exception_____ e) {
        System.out.println("File missing.");
    }
    // ... use scanner
}
```

Problem 13: Reading All Lines

Task: Print every line in the file.

```
// Assume 'sc' is a valid Scanner
while (sc.hasNextLine()______) {
    String currentLine = sc.nextLine()______;
    System.out.println(currentLine);
}
```

Problem 14: Parsing CSV Line

Task: Given a line "Name,Age,Grade", extract the age (2nd item).

```
String line = "Alice,20,A";
// Split into array
String[] tokens = line.split(",")____________;

// Parse the integer Age
int age = Integer.parseInt_________(tokens[_1__]);
```

Problem 15: Clean Up Resources

Task: Ensure the Scanner closes even if an exception occurs.

```
Scanner sc = null;
try {
    sc = new Scanner(new File("test.txt"));
    // ... work ...
} catch (FileNotFoundException e) {
    System.out.println("Error");
} finally {
    if (sc != null) {
        sc.close(); // Close it
    }
}
```

Problem 16: Validating Int Input

Task: Read an integer from user; if not an int, discard token and retry.

```
Scanner console = new Scanner(System.in);
System.out.print("Enter age: ");

while (!console.hasNextInt()__) { // Check ONLY if next is int
    String trash = console.next(); // Discard bad input
    System.out.print("Invalid. Enter age: ");
}
int age = console.nextInt()_____; // Read the valid int
```

Problem 17: String Equality

Task: Check if input command is "QUIT" (case-insensitive).

```
final String STOP_CMD = "QUIT";
String input = getCommand();

// Check equality ignoring case
if (input.equalsIgnoreCase_________(STOP_CMD)) {
    System.out.println("Exiting...");
}
```

Problem 18: Char At Index

Task: Print the first and last character of a non-empty string ‘str‘.

```
char first = str._Char At______(0);
// Get last index based on length
char last = str.charAt(__str.length() - 1______);

System.out.println(first + " " + last);
```

Problem 19: Summing Doubles from String

Task: Sum numbers in string data "1.5 2.5 3.5".

```
String data = "1.5 2.5 3.5";
Scanner stringScan = new Scanner(_data____); // Read from String
double sum = 0.0;

while (stringScan.hasNextDouble()) {
    sum += stringScan.NextDouble()______;
}
```

Problem 20: Manual String Copy

Task: Manually copy characters from string ‘src‘ to char array ‘dest‘.

```
String src = "Hello";
char[] dest = new char[src.length()];

for (int i = 0; i < src.length(); i++) {
    // Get char at i and assign to array
    dest[i] = src.charAt(i)____;
}
```

Long Form - Section Overview

- Write complete classes or methods to solve these problems.
- Adhere strictly to the Style Guide:
  - Class structure(Constants -> Vars -> Constructors -> Methods).
  - No magic numbers.
  - Proper naming conventions.
  - Meaningful comments for logic.

Problem 21: File Line Counter

Write a complete program ‘LineCounter‘ that:

1. Defines a constant for the filename "input.txt".
2. Opens the file using a Scanner.
3. Counts how many lines are in the file.
4. Prints "Count: X" to the console.
5. Handles `FileNotFoundException` by printing "File missing".
6. Ensures the Scanner is closed.

Problem 22: String Reverse

Write a static method ‘public static String reverse(String input)‘ that:

1. Takes a String as a parameter.
2. Returns a new String with characters in reverse order.
3. Uses a standard ‘for‘ loop to build the result (using concatenation or StringBuilder).
4. Handles null input by returning null.

Style Requirement: Do not modify the loop variable inside the loop body.

Problem 23: CSV Parser

Write a method ‘processStudentData‘ that reads student grades from a file.

- Input Format: "Name,Grade" (e.g., "John,85").
- Read line by line.
- Split each line by comma.
- If Grade $>$ 50, print "Name passed".
- Use ‘Integer.parseInt‘ to convert the grade.
- Must use a constant for the PASS_CUTOFF (50).

Problem 24: Valid Input Loop

Write a method ‘getPositiveInt‘ that:

1. Takes a Scanner (tied to System.in) as a parameter.
2. Prompts the user "Enter positive number: ".
3. Loops until the user enters a valid integer AND it is > 0.
4. If input is not an integer, discard it and print "Not a number".
5. If input is ≤ 0, print "Must be positive".
6. Returns the valid positive integer.

Problem 25: Palindrome Checker

Write a method ‘isPalindrome‘ that checks if a string is the same forwards and backwards.

- Ignore Case (treat ’A’ and ’a’ as same).
- Use ‘charAt‘ to compare characters from both ends moving inward.
- Return boolean ‘true‘ or ‘false‘.
- Use meaningful variable names (e.g., ‘leftIndex‘, ‘rightIndex‘).

Problem 26: Max In File

Write a complete program ‘MaxFinder‘.

  - File "numbers.txt" contains one integer per line.
  - Read all numbers.
  - Find and print the maximum value found.
  - If file is empty, print "No data".
  - Handle exceptions appropriately.

- Use ‘Integer.MIN$_{V}$*ALUE‘toinitializeyourmaxtrackerifneeded*.

Problem 27: Name Formatter

Write a method ‘formatName‘ that:

- Input: String ‘fullName‘ (e.g. " jOhN dOE ").
- Trims whitespace.
- Converts the entire string to Lower Case.
- Capitalizes the first letter of each word (assume space delimiter).
- Returns the formatted string (e.g. "John Doe").
- You may use ‘split(" ")‘ and string concatenation.

Problem 28: Word Search

Write a method ‘containsWord‘.

- Parameters: String ‘text‘, String ‘keyword‘.
- Returns ‘true‘ if ‘keyword‘ exists in ‘text‘ (Case Insensitive).
- Do NOT use ‘text.contains()‘. Use ‘indexOf()‘ logic manually or looped substring checking if you want a challenge, OR simply use ‘text.toLowerCase().indexOf(...)‘.
- Ensure ‘null‘ checks for both strings at the start (return false if either is null).

Problem 29: File Copy

Write a program that copies "source.txt" to "dest.txt".

1. Open ‘source.txt‘ for reading (Scanner).
2. Open ‘dest.txt‘ for writing (‘PrintWriter‘).
3. Loop through source line by line.
4. Write each line to ‘dest.txt‘ using ‘println‘.
5. Close both resources in a finally block (or try-with-resources if taught, but standard finally is safer for manual practice).
6. Handle ‘FileNotFoundException‘.

Problem 30: Average from String

Write a method ‘calculateAverage‘ that constructs an average from a formatted string.

- Input: String line = "ID:1234 Marks:80,90,100";
- Extract the part after "Marks:".
- Split by comma to get individual marks.
- Parse each mark to double.
- Calculate and return average.
- Avoid magic numbers (e.g. define prefix "Marks:").