**Computer Language 2022**

**Assignment #1**

**Due: 7/Mar 23:59:59**

**Student ID: 21102052 Name: Lee Jeong-Yun**

**1. Install JDK and IntelliJ on your system. And make a program to print out “Hello, [Your Student ID+Name]!” in the console. Capture the output of your program and paste it in this document.**

**Example) Remove the figure below and add your result**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**2. Provide at least 5 examples of invalid variable names. Also, explain why such variable names cannot be used. You cannot use the examples presented in the lecture slide (page 8).**

**1) int for;**

**-> “for” is one of the Java keywords.**

**2) int #stopWar;**

**-> “#” is one of the special characters except “$” and “\_”.**

**3)int 82korea;**

**-> The name of variable begins with digits, “8”**

**4)int true;**

**-> Boolean literal, “true” cannot be used for name of variable.**

**5)int love peace;**

**-> The name of variable includes a whitespace between “love” and “peace”**

**3. Write a program to print the following string using UNICODE literals (only for symbols). Capture your source code, the output of your program and paste them in this document.**

**source code:**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**output:**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**4. Execute the following program. What is the output of this code? Explain why/how such a result was calculated.**

**public class Hello {  
 public static void main(String[] args) {  
  
 int intValue = 77777;  
 System.*out*.println((short) intValue);  
   
 }  
}**

**Output: 12241**

**Why: intValue는 int형 데이터이다. (short) intValue는 intValue의 자료형을 range가 더 큰 int에서 range가 더 작은 short로 형변환하는 casting을 수행했기 때문이다.**

**int형에서 77777은 4바이트인 00000000 00000001 00101111 11010001으로표현된다. short형 자료형은 2바이트이므로 int의 4바이트 중에서 2바이트, 00101111 1101001만을 할당 받는다. 00101111 11010001을 decimal로 변환하면 12241이 된다.**

**\*Calculator for programmer**

**qword –> 8 byte**

**dword –> 4 byte**

**word –> 2byte**

**byte –> 1 byte**

**5. How to declare a variable for the following case? Write your answer here.**

**- int type variable ‘height’ int height;**

**- double type variable ‘size’ initialized to 0.25 double size=0.25;**

**- double type variable ‘total’ initialized with the sum of the values of ‘height’ variable and ‘size’ variable double total = height + size;**

**- char type variable ‘c’ initialized with ‘a’ char c =’a’;**

**- double type constant ‘BODYTEMP’ initialized to 36.5 final double BODYTEMP = 36.5;**

**\*constant -> All CAPITAL Characters!!!**

**\***

**Object tmp = 67+12.8;**

**System.out.println(tmp.getClass().getTypeName());**

**System.out.println(tmp)**

**>>>java.lang.Double**

**>>>79,8**

**6. what is the result and type of the following statements? Explain the reason as well.**

a) 67+12.8

result: 79.8

type: double

reason: Every integer is basically assigned to int type. So, 67 is int type. 12.8 is floating point type, and the number doesn’t have any suffix like F. Then 12.8 is double type. When Java perform an arithmetic operation with different types of values, the language will convert small type value to larger type value. As a result, int (small size type) is changed to double (larger size type) and the type of result should be double, too. The result value is 79.8 because of simple calculation, sum of 67 and 12.8.

**b) ‘c’+1**

result: 100

type: int

reason: The Unicode of c surrounded by single quotes is \u0063. 1 is just int. Char type has 2 bytes and int has 4 bytes, so char should be promoted to int. The hexadecimal 0063 is 16×6+1\*3=99 in decimal. Therefore, since int type 'c' is 99, 1 is 1, the result value is 100, and the data type is int. Arithmetic operations are performed by operands of the same form. Small types are automatically promoted to large types.

**\*+0063 (hex)-> 99 (decimal)**

**c) 10/3**

result: 3

type: int

reason: Both 10 and 3 are int types. Since it is an operation between int, the resulting value must also be int. The result of the actual division is the floating-point type, but since the int value must come out, only the integer part, that is, the quotient, is output.

**d) 10.0/3**

result: 3.3333333333333335

type: double

reason: 10.0 is double and 3 is int. Since the data types of the two data are different, the result value must come out in double, which is a larger type. Arithmetic operations are performed by operands of the same form. Small types are automatically promoted to large types.

**e) 10 == 9**

result: false

type: Boolean

reason: There is '==', condition statement in this equation. It is necessary to determine whether the equation '10==9' is true or false. Therefore, a Boolean type is needed to present true or false. False is output because 10 and 9 are not the same.

**7. Execute the following codes. Then, you will see eight values printed on the console. For each value, explain how it was calculated.**

public class Ex07 {  
 public static void main(String[] args) {  
 byte b = 127;  
 int i = 100;  
  
 System.*out*.println(b + i);  
 System.*out*.println(10 / 4);  
 System.*out*.println(10.0 / 4);  
 System.*out*.println((char) 0x12340041);  
 System.*out*.println((byte) (b + i));  
 System.*out*.println((int) 2.9 + 1.8);  
 System.*out*.println((int) (2.9 + 1.8));  
 System.*out*.println((int) 2.9 + (int) 1.8);  
  
 }  
}

|  |  |
| --- | --- |
| **227** | **Byte b is converted to int b. Value of b is still 127 because byte is smaller size than int. 100 is also int. 127 + 100 = 227.** |
| **2** | **Operands are int, so the result should be int. The actual result 2.5 is needed to be 2.** |
| **2.5** | **Operation requires same type of operands. There are double type 10.0 and int type 4.**  **Since int is smaller than double, the int type will converted to double type. After that, the result will be coming with double type. So, the result should be double.** |
| **A** | **Hexadecimal is used. 0x12340041 is 0001 0010 0011 0100 0000 0000 0100 0001 in binary. The char type has 2 bytes. 0000 0000 0100 0001 is assigned to (char) 0x12340041. 0000 0000 0100 0001 is 41 in hexadecimal(0x0041) and \u0041 is ‘A’.** |
| **-29** | **127 is the maximum value in byte type. When 1 is added thereto, the intermediate result is the minimum value of the byte type, -128 (1000 0000 in binary). Add 99 to the result and the final result becomes -29. (-128+99=-29)**  **\*1로 시작하는 binary는 음수이다. 1000 0000 -> -128**  **Binary 음수만들기 -> inverse -> decimal +1**  **e.g.,**  **00000101 (5) -> 11111010 (-6) -> 11111011 (-5)**  **00011101 (29)-> 11100010 (-30) -> 11100011 (-29)** |
| **3.8** | **Double type 2.9 is converted to byte type to become an integer 2, and 1.8 maintains the double type. The sum of byte and double is changed to an operation between double types through promotion. The sum of 2.0 and 1.8 is 3.8.** |
| **4** | **Converting 4.7 which is a double type to a byte type representing an integrated type results in 4, excluding the decimal part.** |
| **3** | **int 2.9 + int 1.8.**  **'2.9' and '1.8' are double types. Bytes are data types representing integers. Therefore, adding 2 and 1 from which the decimal portions of each of the two numbers are removed becomes 3.** |