

# CMPU102: Exam I Practice Exam Solutions

1. (20 points) Answer the following questions by circling T (true) or F (false) (*Note: stating your assumptions about the question may influence the correctness of your answer*):
- a) *T or F*: Java provides 8 numerical primitive types. - **true. All the primitive types are stored as numbers**
  - b) *T or F*: When passing arguments to a method, the type of the argument must be assignment compatible with the type of the parameter declared in the method header. - **true**
  - c) *T or F*: When the keyword “void” is used in a method declaration, it means that there will be no value returned by the method. - **true**
  - d) *T or F*: Each Java class type has a set of literal constant values that can be stored as the value of a variable for that type. - **false, Java classes are stored as addresses and there are no literal values for class types.**
  - e) *T or F*: Naming a variable A3xYz would cause a syntax error. - **false**
  - f) *T or F*: Java belongs to the category of computer languages known as “machine” languages. - **false, Java is a higher level object oriented language.**
  - g) *T or F*: The number of vertical spaces (blank lines) between statements in a Java program is irrelevant. - **true, the java compiler ignores blank spaces.**
  - h) *T or F*: Like assembly languages, each line of Java code is translated into a single line of machine code. - **false, even assembly language code can't be translated into a single line of machine code**
  - i) *T or F*: A return statement can be present in any method. - **true**
  - j) *T or F*: Java programs do not need to import the java.lang package in order to use the shorthand notation for classes defined in that package. - **true**
  - k) *T or F*: Naming a variable 8Ba1l would cause a syntax error. - **true**
  - l) *T or F*: Once all syntax errors are corrected, a Java program will execute as intended. - **false, there may still be logical errors and runtime errors.**
  - m) *T or F*: Given the following statements: `char letter = '3'; int number = 8;` the subsequent statement `int result = letter + number;` will return a syntax error. - **false, the statement should return a number.**
  - n) *T or F*: The value returned, if the value of the int variable `count` is 0 and the value of the int variable `limit` is 10, as the result of evaluating the expression `(count == 0) && (limit < 20)` is true. - **true**
  - o) *T or F*: An if decision statement must be followed by at least one `else` statement with which it is paired - **false.**

- p) *T or F*: A single class may contain the variables `number` (an `int`) and `number` (a `double`).- **false, if they are both in the same scope; true, if they are declared in an outer and inner scope.**
- q) *T or F*: The assignment statement has the highest precedence of all operators. - **false**
- r) *T or F*: If the variable `count` is initialized as in part n, the result of evaluating the boolean expression `!(count == 12)` is false. - **false**
- s) *T or F*: A for loop always executes its body at least once. - **false, if the condition in the for loop header is false, nothing in the body is executed.**
- t) *T or F*: Computers do math internally using the octal number system. - **false, they use the binary number system.**

2. (4 points) Explain what is meant by the statement “Java is a portable language.” Explain what is different about a non-portable language such as `c++`.

**Once Java source code is compiled into byte-code, it can be run on any computer that has the Java Virtual Machine installed, no matter the operating system.**

3. (6 points) Name and describe the three kinds of errors that must be corrected before a Java program operates as intended, indicate the stage of program development in which each kind is found (testing, compilation, or execution), and indicate what type of software (if any) discovers the error.

**Syntax errors are found during compilation and errors are found by the compiler. Runtime errors are found during execution and the Java Virtual Machine finds this type of error. Logical errors are found during testing and must be found by the programmer.**

4. (10 points) Write a method called `multEven` that takes an `int n` as input and returns an `int` that is the product of all **even** whole numbers between 2 and `n`. Use a **while loop** in your solution.

```
public int multEven(int n) {
    int i = 2;
    int product = 1;
    while (i <= n) {
        product = product * i;
        i = i + 2;
    }
    return product;
}
```

5. (5 points) Circle 5 syntax errors in the following program and explain why each is an error:

```
1.    \*
2.        Program Problem1
3.    */
4.    class Problem 1{
5.        public static myMethod(String st)
6.        {
7.            i = 5;
```

```

8.         double backUp = 1.0/5;
9.         return backUp
10.      }
11.   }

```

line 1: Not a comment -- should be /\*.  
line 3: Multiline comment never started.  
line 4: Should be no space in class name.  
line 5: No return type in method declaration.  
line 7: Variable i not declared.  
line 9: No semicolon at end of line.

6. (5 points) What is the output of the following code fragment? Assume there are no syntax errors.

```

int i = 5;
int j = 7;
int k = 4;
if (( i >= j ) || (k < 5)) {
    System.out.print("happy");
}
else {
    System.out.print("joy");
}
if (( i != j ) && (k > 4)) {
    System.out.println("happy");
}
else {
    System.out.println("joy");
}

```

**happy joy**

7. (5 points) Give the output of the following code if the call to this method is `PetStore.buyCompanion(0);` (assume there are no syntax errors)

```

public class PetStore {
    public static void buyCompanion(int k) {
        switch (k + 3) {
            case 1:
                System.out.println("Puppy");
            case 2:
                System.out.println("Macaw");
                break;
            case 3:
                System.out.println("Kitten");
            case 4:
                System.out.println("Goldfish");
                break;
            default:
                System.out.println("Cricket");
                break;
        }
    }
}

```

```

    }
  }
}

```

**Kitten  
Goldfish**

8. (15 points) Rewrite the class in the last problem using an if, else if, else multibranch statement instead of a switch to get the same results as the switch statement.

```

public class PetStore {
    public static void buyCompanion(int k) {
        j = k + 3;
        if (j == 1) {
            System.out.println("Puppy");
            System.out.println("Macaw");
        }
        else if (j == 2) {
            System.out.println("Macaw");
        }
        else if (j == 3) {
            System.out.println("Kitten");
            System.out.println("Goldfish");
        }
        else if (j == 4) {
            System.out.println("Goldfish");
        }
        else {
            System.out.println("Cricket");
        }
    }
}

```

9. (15 points) Give the output of each of the following code fragments. If you think any of these loops would execute infinitely, give only the first few iterations of output and indicate that the execution is infinite. If you think there would be no output, write “no output.”

---

a) (3 pts)

```

int count = 40;
for (count = 1; count < 5; count++);
System.out.println("count is "+(2 * count));

```

**count is 10**

The for loop only contains a “do nothing” statement ; but the value of count is 5 when the loop ends.

---

b) (3 pts)

```

int n = 0;
for (int n = 10; n >= 0; n -= 2)
    System.out.print("Hello " + n);

```

**Hello 10Hello 8Hello 6Hello 4Hello 2Hello 0**

---

c) (3 pts)

```
int number = 10;
while (number > 10);
{
    number = number + 5;
    System.out.println(number);
}
```

15

---

d) (3 pts)

```
int number = 10;
while (number >= 0)
{
    number = number - 2;
    if (number == 4)
    {
        break;
    }
    System.out.println(number);
}
System.out.println("The end.");
```

8

6

The end.

---

e) (3 pts)

```
int number = 11;
do
{
    number = number - 2;
    if (number == 4)
    {
        break;
    }
    System.out.println(number);
}while (number > 0);
System.out.println("The end.");
```

9

7

5

3

1

-1

The end.

---

10. (15 points) For each of the parts a-e, below, write a statement that is a call to method bottomLine with literal boolean arguments for boolean parameters a, b, c, and d that produces the output indicated.

```
public void bottomLine(boolean a, boolean b, boolean c, boolean d){
    if (a && b)
    {
        if (!c && !d)
```

```
        System.out.println(1);
    else if (!d)
        System.out.println(2);
    else
        System.out.println(3);
}
else if (c==d)
    System.out.println(4);
else if (c)
    System.out.println(5);
else
    System.out.println(6);}

```

Note that there may be more than one possible combination of truth values for the four boolean parameters that produce a particular output. You are only required to write a method call containing one such combination of literal boolean values for each of parts a) through e) below.

- a) (3 pts) Write a call that causes the method to display 1 to the standard output.  
**a = true, b = true, c == false, d = false**
- b) (3 pts) Write a call that causes the method to display 2 to the standard output.  
**a = true, b = true, c == true, d = false**
- c) (3 pts) Write a call that causes the method to display 3 to the standard output.  
**a = true, b = true, c = false, d = true**
- d) (3 pts) Write a call that causes the method to display 4 to the standard output.  
**a = false, b = true, c = false, d = false**
- e) (3 pts) Write a call that causes the method to display 5 to the standard output.  
**a = false, b = true, c = true, d = false**