

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Preparing for Object Oriented Programming

PDF generated at 15:42 on Monday 14th August, 2023

1.1P: Preparing for OOP – Answer Sheet

1. Explain the following terminal instructions:
 - a. cd: **Change Directory** (change the current working directory in the terminal)
 - b. ls: **List** (list the files and directories in the current directory)
 - c. pwd: **Print Working Directory** (navigating and managing files and directories in a terminal environment)
2. Consider the following kinds of information, and suggest the most appropriate data type to store or represent each:

Information	Suggested Data Type
A person's name	String
A person's age in years	Integer
A phone number	String
A temperature in Celsius	Float
The average age of a group of people	Float
Whether a person has eaten lunch	Boolean

3. Aside from the examples already provided in question 2, come up with an example of information that could be stored as:

Data type	Suggested Information
String	Residential Address
Integer	Count of Inventory
Float	Person's Height in meter
Boolean	Was it rain today

4. Fill out the last two columns of the following table, evaluating the value of each expression and identifying the data type the value is most likely to be:

Expression	Given	Value	Data Type
------------	-------	-------	-----------

6		6	Integer
True		True	Boolean
a	a = 2.5	2.5	Float
1 + 2 * 3		7	Integer
a and False	a = True	False	Boolean
a or False	a = True	True	Boolean
a + b	a = 1 b = 2	3	Integer
2 * a	a = 3	6	Integer
a * 2 + b	a = 2.5 b = 2	7.0	Float
a + 2 * b	a = 2.5 b = 2	6.5	Float
(a + b) * c	a = 1 b = 1 c = 5	10	Integer
"Fred" + " Smith"		"Fred Smith"	String
a + " Smith"	a = "Wilma"	"Wilma Smith"	String

5. Using an example, explain the difference between **declaring** and **initialising** a variable.

The difference between the two is based on their respective functions and actions within a program.

Declaring a variable includes stating its name and data type to the compiler or interpreter. Which tells the system of the variable's existence and type, allowing memory to be reserved for it.

Initialising a variable consist of giving it an initial value at the point of declaration or subsequently in the program. This gives the variable a specific value, and the variable now carries useful data.

6. Explain the term **parameter**. Write some code that demonstrates a simple of use of a parameter. You should show a procedure or function that uses a parameter, and how you would call that procedure or function.

A parameter is a variable in a function or procedure definition that allows values to be passed into the function when it's called. Parameters define the input that a function or procedure expects to receive, allowing the function to perform its task with specific data.

Example:

```
using System;

class Program
{
    static double CalculateRectangleArea(double width, double height)
    {
        return width * height;
    }

    static void Main()
    {
        double width = 2;
        double height = 4;

        // Calling the function and passing 'width' and 'height' as arguments.
        double area = CalculateRectangleArea(width, height);

        Console.WriteLine("Rectangle Area: {0}", area);
    }
}
```

7. Using an example, describe the term **scope** as it is used in procedural programming (not in business or project management). Make sure you explain the different kinds of scope.

Scope is the visibility and accessibility of variables within different parts of a program. It defines where a variable can be accessed and controlled. There are two main types of scope in procedural programming, local scope, and global scope.

Example:

Local Scope (boy and girl):

```
void total student()
{
    int boy = 12;
    int girl = 9;

    int total = boy + girl;
    Console.WriteLine("Total student: " + total);
}
```

Global Scope (StudentEachClass):

```
int StudentEachClass = 31;
void TotalStudent()
{
    int NumOfClass = 7;
    int total = StudentEachClass * NumOfClass;
    Console.WriteLine("Total Student: " + total);
}
```

8. In a procedural style, in any language you like, write a function called Average, which accepts an array of integers and returns the average of those integers. Do not use any libraries for calculating the average. You must demonstrate appropriate use of parameters, returning and assigning values, and use of a loop. Note — just write the function at this point, we'll use it in the next task. You shouldn't have a complete program or even code that outputs anything yet at the end of this question.

Using C#:

```
using System;

class Program
{
    static double Average(int[] nums)
    {
        int sum = 0;

        foreach (int num in nums)
        {
            sum += num;
        }

        double average = (double)sum / nums.Length;
        return average;
    }
}
```

9. In the same language, write the code you would need to call that function and print out the result.

```
using System;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Text;

class Program
{
    static double Average(int[] nums)
    {
        int sum = 0;

        foreach (int num in nums)
        {
            sum += num;
        }

        double average = (double)sum / nums.Length;
        return average;
    }

    static void Main()
    {
        int[] values = { 1, 2, 3, 4, 5 };

        double result = Average(values);
        Console.WriteLine("Average: " + result);
    }
}
```

10. To the code from 9, add code to print the message "Double digits" if the average is above or equal to 10. Otherwise, print the message "Single digits". Provide a screenshot of your program running.

```
using System;
using static System.Runtime.InteropServices.JavaScript.JSType;
using System.Text;
```

```
class Program
```

```
{
```

```
    static double Average(int[] nums)
```

```
    {
```

```
        int sum = 0;
```

```
        foreach (int num in nums)
```

```
        {
```

```
            sum += num;
```

```
        }
```

```
        double average = (double)sum / nums.Length;
```

```
        return average;
```

```
    }
```

```
    static void Main()
```

```
    {
```

```
        int[] values = { 1, 2, 3, 4, 5 };
```

```
        double result = Average(values);
```

```
        Console.WriteLine("Average: " + result);
```

```
        if (result >= 10)
```

```
        {
```

```
            Console.WriteLine("Double digits");
```

```
        }
```

```
        else
```


```
        {
```

```
            Console.WriteLine("Single digits");
```

```
        }
```

```
    }
```

```
}
```

 class System.String

Represents text as a sequence of characters.

```
1  using System;
2  using static System.Runtime.InteropServices.JavaScript.JSType;
3  using System Buffers.Text;
4
5  class Program
6  {
7      static double Average(int[] nums)
8      {
9          int sum = 0;
10
11          foreach (int num in nums)
12          {
13              sum += num;
14          }
15
16          double average = (double)sum / nums.Length;
17          return average;
18      }
19
20      static void Main()
21      {
22          int[] values = { 1, 2, 3, 4, 5 };
23
24          double result = Average(values);
25          Console.WriteLine("Average: " + result);
26
27          if (result >= 10)
28          {
29              Console.WriteLine("Double digits");
30          }
31          else
32          {
33              Console.WriteLine("Single digits");
34          }
35      }
36  }
37
```

> Terminal – 1.1P 8

Average: 3
Single digits