<u>3 Concepts & Basics</u>

3.1 Explain the differences between a compiled and an interpreted programming language. (2 marks)

A compiled language means that the language is directly converted into machine code that is executed by the processor. An interpreted language runs each command line by line and executes also line by line. A compiled language is usually faster and more efficient than an interpreted language, however it needs to be compiled manually, they need to get rebuilt every time a change is made. Another difference is that a compiled language has more control over the hardware of the machine such as CPU.

3.2 Why Java is secure and platform-independent? (2 marks)

Java produces bytecode which can be readable by many machines and not based on a certain one. So when Java runs in a machine, there is a compiler that converts the code into bytecode, something more readable, this gets sent to the Java virtual machine existing in the RAM of the OS. Then the Java virtual machine recognized the platform and coverts the bytecode into machine code, that is why Java is secure and platform independent.

3.3 Explain the differences and similarities between variables and constants. (2 marks)

A constant is a fixed value and only one copy of it exists in the program. This value cannot be changed throughout the program as it is fixed. Variables are not fixed values and they can be changed. The similarity is that they both hold values, also however, if you use the keyword final before the variable, it turns into a constant value that cannot be changed, which make them very similar.

3.4 What are the 8 primitive data types in Java? What are the differences between the types? For each type (separate lines), give a Java statement creating a variable and assigning a value. (6 marks)

The 8 primitive data types are: Byte, short, int, long, float, double, boolean, char.

Byte: It is an 8-bit integer, but it only holds values from -128 to 127, this can be used in large arrays where every part of memory matters

```
byte small;
small = -120;
```

Short: This is a 16-bit integer, meaning it is larger, and holds values from -32,768 to 32,767, also can be used in large arrays if memory matters but you need a larger value

```
short kilometre;
kilometre = 170;
```

Int: A 32-bit integer that holds values from -2^31 to 2^31-1, this would mainly cover all numbers unless extremely, extremely large.You can use this for unassigned integers, too.

```
int large;
large = -5600000;
```

Long: A 64-bit integer that holds values from -2^63 to 2^63-1. This is for values even larger than int.

```
long distance;
distance = 432346642L;
```

Float: It is a double-precision 32-bit floating point, that should not be used for precise values. The range of values for a float is too extreme to discuss. You can use this instead of double to save memory.

```
float num;
num = 67f;
```

Double: Similar to float, but it is a double-precision 64-bit floating point that should never be used for precise values like currency

```
double number;
number = 45.3;
```

Boolean: There are two possible values, ‘true’ and ‘false’.

```
boolean situation = true;
```

Char: This is a 16-bit unicode character. The minimum value of a data type being \u0000 and maximum of \uffff.

```
char letter;
letter = ‘u0042’;
```

3.5 What is casting? Explain the differences between implicit and explicit casting. (2 marks)

Casting is changing the value from one primitive data type to another.

Implicit casting, also known as widening, is when we assign a smaller value to a larger value, when they are both compatible.

Explicitly casting, on the other hand, known as narrowing, is the opposite, moving from a larger data type to a smaller one, only when the values are compatible.

3.6 What is overflow? Please, give an example of overflow. (2 marks)

Overflow happens when we assign a data value that is out of range for the primitive data type being used. So if the absolute value is too large, we call it overflow. An example is if we try to assign 1000000 to a byte which only covers from -128 to 127.

3.7 What are the four main features of Object-Oriented Programming? Please, give a definition of each key feature. (4 marks)

- Encapsulation
    - The different objects in each program will communicate with each other. If the programmer wants to stop this from happening, he needs to place objects is separate classes
    - Each object has a barrier encapsulating it from other objects that it should not interfere with
- Abstraction
    - This is a follow-up from encapsulation with a different perspective, if a mistake has been done in one class, it stays in this class and will not affect other classes
- Inheritance
    - This feature stops repetitive code from happening. Instead of repeating code for more than one class, it can just inherit it from the superclass, and the objects that follow would be subclasses that adopt those features from the superclass
- Polymorphism
    - The meaning behind this is that the classes can take any shape they want. The subclasses can adopt from the parent classes while keeping their own methods. Each subclass or child class can implement their own way of using the parent class. For example, the parent class being able to make figures or shapes, and the child classes being triangle, rectangle, and square.