Talking OOP Java. Version Two

You must express yourself using the correct Java terminology.

Data and Information

- **Data** is either primitive or complex. There are “primitive data variables” (int, double, boolean, char) or “complex data structures” (classes and objects – String is complex). Raw data is not useful until it is processed.
- **Information** is processed, structured, organized or filtered and is therefore useful.

Data that is read out of a text file is raw data because it is not organized (the field names are not part of the text file usually)

Classes and Objects

- A class is a template from which we create child objects.
- An object is an instance of a parent class.
- From a class we create instances of that class (instantiation)

We **instantiate** an object using the parent class as a template – the created object inherits the data and methods of the parent class but is its own separate entity. Classes do not **contain** objects – they define the data and methods that the child object will have.

A newly create object is also a class - you can instantiate a new child object from it. The new child object will inherit all data/methods from both parent classes.

Classes are **encapsulated** (an advantage of OOP programming). The data inside an object is protected from outside interference –therefore the primitive data types in the class are declared as private.

We should only use the **accessor methods** and **mutator methods** to alter the values inside a class.

Classes have methods - 1) constructor method(s) 2) getter methods (accessor methods) 3) setter methods (mutator methods) 4) toString method.

Constructors

You instantiate a new object from a class using the special class method called a **constructor.** A class can have more than one constructor – we say that the constructor method is **overloaded**. Note: All the constructors have the same name i.e. the **same name as the class** but have a different number of parameters. See below.

<u>Constructor and parameters</u>

- One constructor can accept no parameters.
- Other constructors can accept one parameter.
- A third constructor can accept three parameters etc.
    - Two constructors cannot have the same number of parameters

All constructors within a class have the same name as the class.

All classes have a constructor method, even if you don't create one - Java will provide a default one for the application to use if relevant. The default constructor will instantiate the object with default values in all fields e.g. int will be 0, double will be 0.0, String will be null, boolean will be false, char will be the very first ASCII value ('\u0000)

Java knows which one to use as it matches the arguments in the call statement with the parameters in one of the constructors – a call with three arguments is matched to the constructor with three parameters.

> Therefore when we instantiate new objects they can there have different properties because they were created using different constructors (one vehicle object can have the colour property "white" while another has a null value in the colour field – note – all will have the colour field but the data in the colour field can vary from null to any colour you can think of.

toString Method

- If you don't create your own toString method Java will create one for you – a default method. It will concatenate *all* the fields together in the same order that they were declared in.
- If you create your own toString method you can choose the fields you want, their order and format.

Methods are declared private when they are helper methods – they are only needed internally by that class (or any child class created from it.)

Math.pow(double a, double b) is a public method of the Math class. It can be called anywhere at any time. It is static because we never need to create a child object from the Math class.

A class has data/properties and methods. We create instances of the class, which become separate child objects – we use the constructor method to do this. Instance objects inherit all the data and methods of the parent class.

Methods in a class can be void or typed. A void method returns nothing e.g. it changes the colour from red to blue. A typed method returns a value
e.g. price * VAT – we need the new value back.

Arrays and an arrays of objects

- An array is a complex data structure.
- An array is "static". Different meaning. The structure of an array may not be altered after it is created using the new keyword e.g. new Puzzle[200].

private Puzzle[ ] puzzleArray = new Puzzle[200]

> This array of 200 is indexed from 0 to 199 (it has 200 elements)
> The size of the array may not be altered – it is "static"

An array can only contain elements of the same data type i.e. the same primitive data type or the same complex data type. NOTE: Don't be fooled. An array of objects still contains the **same** data type – in this case it is the same complex data type (that contains many primitives variables of different types.)

Class diagrams
Using puzzles in a toy store; they would not only have one – therefore the full program would create an **array of puzzle objects**.

Field/Properties – all private
1. Name of the puzzle
2. Description of the puzzle
3. Level of difficulty – 1 to 10
4. Price
5. Imported – yes or no

NOTES:
The method "updatePrice" is a setter (mutator method). You do not have to use the word "set".
The parameters and their primitive data types are part of the class diagram.
- **Without brackets** is a return type - getName : string
- **With brackets** is a parameter that is passed - updatePrice (p : double)
- The method getWelcomeScreen – no parameters and no return type

Puzzle
Properties:
- name : string
- description : string
- difficulty : integer
- price : double
- imported : boolean
Methods: (just some possible examples)
+ Constructor (n : string, de : string, di : integer, p : double, im : boolean)
+ Constructor (n: string, d: string)
+ getWelcomeScreen
+ getName : string
+ setDescription (d: string)
+ updatePrice : (p : double)
+ toString : string

Method Headers – Four parts
1) Access modifier, 2) return data type, 3) name of method, 4) parameter list (can be empty) – these four are compulsory**
- public String getName(int i) – method needs the index number of puzzle
- public void setDescription(string d)

** public Puzzle (String n, String d) - *Constructor method. No return type.*