HOMEWORK WEEK 3

This week's homework is a research based one. You'll need to conduct independent learning, in combination with existing material (where available), to answer the questions below. The reason for this homework is to ensure you are aware of critical topics in CS. These topics were difficult to cover within the existing lesson schedules, but due to their importance are placed within the homework instead. Make sure to research, learn and then answer the following:

1. What is OOP? How may you have already made use of it (e.g. class components)?
    a. *Feel free to give a fairly light answer here - as you'll need to do the deep-part / actual meat in the following questions when you cover each of OOP's pillars*

"Programming paradigm" is a term used to classify different programming languages, and OOP (object-oriented programming) is one of the most widely used classifications. Python, Java, JavaScript, and Ruby are examples of object-oriented languages. In OOP, code is structured using classes and objects. A class is a broad category used to create objects, which are instances of the class. Examples of objects within classes are strings, integers, and lists. For example, if furniture was a class, then desk, table, and chair would be objects within the class. Classes and objects can also have attributes. Class attributes are characteristics shared by every object in the class, such as material = wood for the furniture example. Object attributes are characteristics belonging to individual objects, such as price. Components are reusable pieces of code. In React, components are independent, reusable, elements which can be made using either classes or functions.

2. What is Polymorphism?

Polymorphism is the concept of data having the ability to occur in different forms depending on the scenario. For example, it allows a child class to override a method from its parent class. Another example is how the + sign can be used in different ways, for example adding integers or concatenating strings, or how a cursor can change from a pointer to a vertical line when hovering over text. Polymorphism can be dynamic, where the polymorphism occurs at runtime, or static, where polymorphism occurs at compile time.

3. What is Abstraction?

Abstraction is hiding details in code which don't need to be understood and only displaying necessary information, enabling complex logic to be used while knowing what the code does, but not necessarily how it works.

4. What is Inheritance?

Inheritance is when a child class derives from a parent class, acquiring the attributes and methods of the parent as well as being able to have its own. For example, if the parent class is plant and an attribute is green, the child class could be cactus, and as well as inheriting the green characteristic it would also have its own individual attribute – spines. The child class is an extension of the parent.

5. What is encapsulation?

Encapsulation is binding data and methods or functions which work on that data together within a singular unit such as a class. It also protects data by hiding it from other classes, as it is only

accessible with the methods/functions within the unit it is enclosed in. This keeps the data and methods together and makes the application more secure.

6. What is:
    a. Agile development?

A methodology in which software development projects are broken down into phases. The most important features are decided on and prioritised and specific work is set to be completed within a dedicated period, and after that the next most important are worked on and so on. This means that teams deliver work in small increments continuously. This enables plans to be changed, and changes can be made depending on how requirements change as the project goes on, due to feedback or changes in the market. A timeline does not necessarily need to be established in advance and can be determined as the project goes on. Development and testing are concurrent, meaning that bugs can quickly be identified and worked on.

    b. Waterfall development?

Waterfall development is a linear methodology, where the next phase of a project isn't moved onto until the previous phase is complete. The requirements are established before product development begins. Then the software is designed and constructed. Testing doesn't start until the development is complete. After that the software is tested and debugged, followed by implementation.

    c. How do they differ? Which is suited for which situation?

Waterfall is more traditional and better suited for projects with definite requirements and concrete timelines. There is a set timeframe and at the end the product should be fully finished and tested. It's easier to manage, however it's more difficult to move back if a previous phase needs to be changed.
Agile is adaptable depending on feedback, updates, and changes in the market, and more flexible with timelines. It allows more communication between everyone involved in the process, and the client is continuously involved. It is more complicated to manage than Waterfall.
Agile is better suited to projects where the timeline may be difficult to determine in advance and where the requirements/specification may change, such as new products. Waterfall is more suited to projects where the constraints are already well understood, the client may not need to be as involved, and there is a rigid specification for the product which is not likely to change.

Once complete, please return to your instructor your answers! Remember:

- Justify and be critical of everything! This distinguishes a great answer from a good answer
- Analyse - why does x even exist? Who needs it or uses it? Who is it important to, what's the point of it at all?