What are the problems addressed by OOP?

Procedural / structured programming approaches were created around the 1970's and involved the creation of a collection of procedures for solving a problem. Once the procedures were determined, the next step was to find appropriate ways to store the data. Hence it is said that the algorithms came first for manipulating the data, and the data structures for storing and making the manipulations easier, came second. Object orientated programming however reverses the order, by putting data handling first and then the algorithms to operate on the data second. (Hortsmann & Cornell, 2012)

For small problems, the procedural approach is said to have worked well. But an object orientated approach is more appropriate for larger scale problems and software. This came down to the ability for the object orientated approach to tackle important economical and methodical issues.

Demand for new and more powerful software had been increasing as the capability of technology increased. Computers / machines became not just more powerful but also included more features; they were becoming more and more sophisticated and connected. This meant an increase in the demand not just for software that matched the capabilities of the hardware, but also from customers of software who began to see the potential and applications of it in a wide range of industries.

Building such software quickly, accurately, cost effectively and with good security measures, was an important goal for teams of software developers to meet. Using the object orientated approach helped to meet these requirements much more than effectively than procedural programming did.

How are they addressed?

Modularity and Reusability:

The nature of Object orientated programming allows us to utilise modularity and reusability in order to build larger software systems much more easily.

We can think of classes and the objects created from them as a basis for this. In OOP, we use classes to define the blueprint of an object (i.e. what are its properties / attributes). In a class, we also define methods that contain program statements about the tasks to be performed with this object. An object is then created as an instance of its class, and we can create many objects from a class.

Now that we have a blueprint of an object, we can reuse the classes and methods within them as reusable software components. This is also applicable through existing classes such as the math library in Java which allows one to make mathematical operations, without having to know the insides of the class that define them (Deitel & Deitel, 2012). This also means we can reuse code without having to type it all out again. The technique for this is called inheritance in object orientated programming and saves the duplication of work.
It also helps to keep code simpler and cleaner for the programmer to read. If the code does require changes, it means that the programmer does not need to change it in several places, but in just one place, as the items referring to this code will simply inherit the changes alike. This improves not just the ability to create clean simple code, but to maintain it also. (Kerber, 2018)

Furthermore, if we have a group of objects from different classes that we would like to do the same operation on, we can define an interface for these classes which allows one to compare them easily. We can also define functions if we want to carry out the same operation of a multitude of classes and objects.

By splitting a problem into classes and methods, we introduce an abstraction of modularity that procedural programming would not have, and this makes OOP much easier for a programmer to grasp and dissect. For example, a simple web browser may require 2000 procedures for its implementation, all of which manipulate a global set of data. However OOP may need 100 classes with an average of 20 methods per class. Not only is the structure of the programme and what it is doing easier for the programmer, but it also easier to find bugs in. If we suppose that the data of a particular object is an incorrect state, it is much easier to search for the culprit among the 20 methods that had access to that data item in a class, than 2000 procedures in one go.
(Hortsmann & Cornell, 2012)

Encapsulation providing security:

In classes, we encapsulate the properties and methods into the objects themselves. Although objects can communicate with each other, they are not usually able to know how another object is implemented, as the details of the implementation itself are hidden with the object. This is useful for security in software engineering because we cannot find out information we should not be privy to such as a password. We only the know object user but the not the user's password encapsulated in the object. (Hortsmann & Cornell, 2012)

Design before coding:

The nature of OOP for large scale systems encourages you to approach and segment the problem into smaller problems, before you just sit down and write code. This sit down and go approach would have been fine for procedural approaches of smaller problems but it is not economical for large scale problems. The approach of design before coding using an object orientated approach enables you to implement big software systems as teams easily, mainly using the Unified Modelling Language.
(Deitel & Deitel, 2012)

What are the remaining issues of OOP?

Object orientated programs tend to be larger than programs using other techniques, where you only wrote what was necessary. This means they take up more memory. It also implies they can be slower than other programs to run because of their size, and inherently they could demand more resource from the computer or syste. Furthermore, a lot more effort might be necessary to create an OOP program, because some significant planning and abstraction is required before code can be written.
When working in an OOP environment, there can be a time when one has to apply OOP principles to a small trivial project or problem, where the complexity involved with this inherently unnecessarily. We may want to do something very simple, but we have to structure everything in an object orientated way so that it works with the rest of the software system.

Bibliography:

Deitel, P. & Deitel, H., 2012. *Java How to Program - Ninth Edition.* s.l.:Pearson. pg47

Hortsmann, C. S. & Cornell, G., 2012. *Core Java.* 9th Editions ed. s.l.:Prentice Hall. pg126

Kerber, M., 2018. *Softwore Workshop - Pages - Teaching - Term 1.* s.l.:University of Birmingham. Lecture 6 - Slide 4 of 9.