**Lecture 1**

OOP: driven by modeling the code around objects

Objects have Fileds, and are capable of performing actions (methods)

Primitive variables – primitive datatypes e.g int,float etc

Objects are enhanced datatypes

**Fields**

The fields of an object are all the data variables that make up that object. They are also sometimes referred to as **attributes** or **member variables**.

These fields are usually made up of primitive types like integers or characters, but they can also be objects themselves.

For example a book object may contain fields like title, author and numberOfPages.

Then a library object may contain a field named books that will store all book objects in an array.

**Accessing fields:**

Accessing a field in an object is done using the dot modifier ‘.’

For example, if we had an object called book that contains these fields:

String title;

String author;

**int** numberOfPages;

To access the title field you would use

book.title

This expression is just like any other string, which means you can either store it in a string variable:

String myBookTitle = book.title;

Or use it directly as a string itself and perform operations like printing it:

System.out.println(book.title);

**Setting Fields**

You can also change a field’s value. Say you want to set the number of pages in a book to 234 pages:

book.numOfPages = 234;

# Methods

You might have also noticed that running actions in objects look very much like calling a function. That’s because that’s exactly what it is.

Methods in Java are functions that belong to a particular object. When we get to creating our own object types later in this lesson we will be creating methods the same way we used to created functions.

### Calling a method

To use a method you call it (just like calling a function). This is also done using the dot modifier .

Methods, just like any function can also take in arguments. For Example: Assume that our book object has a method called setBookmark that takes the page number as a parameter:

**void** **setBookmark**(**int** pageNum);

If you wanted to set a bookmark at page 12, you can call the method and pass in the page number as an argument:

book.setBookmark(12);

## Summary

**Fields** and **Methods** together are what make an object useful, fields store the object's data while methods perform actions to use or modify those data.

However some objects might have no fields and are just made up of a bunch of methods that perform various actions.

Other objects might only have fields that act as a way to organize storing data but not include any methods!

## Next Step

Now that we’ve seen how to use objects and access their fields as well as call their methods, let’s set up your computer so you can start using objects straight away.