CS2

Object Oriented Programming in Java

Handout : Week1

prepared by N S kumar

Preface

Dear friends,

This is an attempt to make Lecture Notes for the course titled CSII Theme: Object Oriented Programming in Java". A team of teachers as well as a few of your seniors are helping in making these documents.

We are trying to highlight the concepts in programming and in Object Orientation. We have observed that many concepts are wrongly understood in academics and the industry. We are trying to make this course simple, interesting as well as challenging. We would appreciate your suggestions in making this course successful and making these handouts better.

We are providing external links to satisfy your curiosity. These contents shall be part of your evaluation process.

We are now giving you the first set of lecture notes covering the first week of lectures. Enjoy.

You may send your feedback to the email id listed below.

N S Kumar
Visiting Professor
Dept. Of Computer Science and Engineering
PES Institutions.
Email : ask.kumaradhara@gmail.com
Date : 11th Jan 2015

Introduction:

I am Kumar. I have been teaching at PES institutes for over 4 years. I stay at Jayanagar. I have to commute between Jayanagar and PES on the days I have to be at PES. Most of the times, I come by car - lazy that I am, I depend on my driver. If my driver does not turn up for whatever reason he may have (may be, his dog developed high temperature, cat ran away, ...), I will travel by autorikshaw. Sometimes college arranges a cab. Once in a while, I travel by BMTC buses. I can also walk – not far – just over 5 km – but there is little path for pedestrian to walk.

So, I have a choice – rather number of them. I can choose the vehicle in which I travel. Commuting is one of the problems we face in the real world. There are lots of others – like preparing for a test, writing an assignment. Every problem provides a variety of possible solutions. This is what Dijkstra observed way back in 1960s. One preparing for a test may read the notes of the teacher or read some standard books or hunt for the question paper. I am sure you would not choose the last option.

Once I decide the mode of transport, I still have a choice in a few cases. Which route shall I take? Shall I try to prove that the world is round? Shall I take the right royal path? Shall I avoid signals and go through narrow lanes – you could be deadlocked by traffic from other sides.
The point I am trying to make is

There is no single solution to a problem

We use computers to solve problems – evaluate assignments automatically, grade the students (may be favour the students!), generate the seating arrangement for tests, randomly choose questions for a quiz,...

We can try to solve problems using different approaches. Each approach indicates a different way of thinking. All of them may lead to the same solution, but in different ways. My aim is to reach the college no matter which mode of transport I take.

There are different paradigms in problem solving

Some of the well know paradigms are procedural, object oriented, object based, generic, functional and logical. We do use sometime combinations of these paradigms depending on whichever suits our scheme of solution. There are many computer languages which have been developed to suit one of these paradigms. Prolog supports logic programming. Haskell supports functional programming.

In this course, we have made a deliberate decision to use object oriented paradigm for designing our solutions. It is the most popular paradigm used in the industry today. Once we select a paradigm, we can discuss solutions to problems at a very high level. For this to be useful, we will have to provide the implementations in some programming language.

Here again, we have choices. Smalltalk, Simula, Modula 2, Java, C#, C++.... The most popular object oriented language in use today is undoubtedly Java. So we have selected Java for you..

That is how we come to this course.

CS2:
Theme: Object Oriented Programming in Java

I am typing this article on my laptop – which is a computer running an operating system called ubuntu.
I am using word processing software called LibreOffice.
I am sitting on a chair and my computer is on the table – not my lap. Around my

computer, I can see lots of books, lots of papers scattered around. My wrist watch is besides my computer. There are a few pens around – some with no caps. Some of the books around me are a bit far for my reach. I have to get up from the chair to pick the book. These books have lots of pages. I can open the book, go to a particular page and try to read the content (Or skim through the way babies do) or may tear the page.

May be, I should arrange the books as computer science books, general books, exercise books(Is that the right term?). I can also classify books based on the language of writing.

Around me, everywhere I see books, pens, chairs, table, computers, ... All are objects. These objects can interact, can react (touch touch-me-or-not, pull tiger's tail). These objects fall into a class of categories. I can count the number of pages in a given book amongst all books I have. I can also count the number of books.

When everything around me can be grouped into class of objects exhibiting similar characters and objects exhibiting individual characteristics, why not use these ideas in solving problems in computer science. Thats how the Object oriented paradigm was born.

Frequently Asked Questions:

Let us have a look a few questions you may have about this course.

Q1. Why should we learn programming at all? Many of us will be studying for a degree in a non-Computer Science branch.

Computers are required in every walk of life – is required in every branch of Engineering. You may use the computers to analyze electric circuits. You may use it to plan buildings, design dams and power stations, analyze protein sequences, compute fluid dynamics. This is a very small set of applications in some Engineering domains. Computers are still dumb – have to be instructed to do what they should. We can tell them what to do only by writing programs – we cannot

shout at them in English and/or Kannada (or the police language!) so far. So we have to learn programming.

- Programming is all about thinking – makes us smarter and intellectually sharper.
- Engineers with good knowledge of programming get better placement than others.
- Good knowledge of programming helps you in projects and research activities in your domain – therefore your chance of getting a seat for masters in a good University increases.
- Most importantly (for me!), you can become lazy. So you will get more time to watch movies or play football! It is said that the very good programmers are the laziest people on Earth!

Q2. Already I know programming. Why should I go through this course?

I am sure all of you are very intelligent and sharp. I am sure that many of you have learnt programming in your earlier schooling. Then you can use this course for the following:

- You may validate whether your learning was in the right direction. There are many books which do not teach programming the right way. In fact I am learning how teach programming for the last 15 years and I am still learning.
- You may help your friends learn programming better. You learn better by teaching others. It does not mean that you hand over copy of solutions to assignments to your friend. In that case you are not helping your friend – you should try to assist your friend in thinking and evolving a good solution.
- I strongly believe that learning is never complete. There will be always something to learn. If you approach this course with this open attitude, it would definitely help you.

Q3. We are learning 'C'. Some of us know C++. Why should we learn Java?

Bjarne Stroustrup who was here last week keeps saying that an Engineer should know at least three different ways of programming. Java is one of widely used Object Oriented Programming Language. Each language has its own way of thinking – has its own philosophy. Programming in a language requires that we develop a taste for that language. (Masala Dosa in VidyarthiBhavan is not same as the one in Woodys on Commercial Street).

You may want to have a look at this video.
“You Should Learn to Program_ Christian Genco at TEDxSMU”
https://www.youtube.com/watch?v=xfBWk4nw440

I will assure that we will make the course interesting and challenging for all of you. Enjoy the course as it unfolds. All the best.

Evolution of Programming Languages:

In the beginning computers had to be instructed by writing programs in the machine language. Writing programs in machine language was tough. Finding the bugs and removing them was also tough. Removing errors in a program is called debugging for historical reasons. You may want to check this link to find the photograph of a dead moth which had short circuited a relay in an old computer.

http://www.jamesshuggins.com/h/tek1/first_computer_bug.htm

These programs would work only on that particular computer as each computer spoke and understood different languages. We could not make the programs portable – take from one computer to another.

John Backus created a high level (away from the computer) called **fortran** – standing for formula translation. He designed a program to convert program in fortran to a machine language program – we now call that a compiler. This was in 50s. He showed to the world that a program in a high level language can be as good as a program in machine language. He also showed that these programs are portable as they do not depend on any particular computer.

Fortran was designed for Scientific and Engineering Community. It was very good in number crunching. With the idea of using computers in the world of business, a language called **COBOL –** COmmon Business Oriented Language – was designed by a group called CODASYL – Conference of Data System Languages. This language is very good in handling huge amount of data, but performs very simple operations on them.

Great Computer Scientists and Academicians joined together and developed a language called **ALGOL –** ALGOrithmic Language – this could assist in writing good programs.

IBM – the biggest company in the field of computer even today – came with an idea of a Universal Language – which would have features of all these languages and something more. This is **PL/I** – read as Programming Language One. This language became a monster – too big – and nobody could master it. So it failed.

Pascal was designed by Prof. Niklaus Wirth to teach programming. Its strength was (should I say is) simplicity. Pascal became a standard for comparison for number of years.

'C' was designed by Dennis Ritchie at Bell Labs of AT & T. It was designed for softwares which talk to hardwares – such programs are called system softwares. These programs are expected to work fast. Efficiency of program execution became the main goal of 'C' even at the cost of safety.

Simula was designed Ole-Johan Dahl & Kristen Nygaard. This was the first language which supported the Object Oriented Concepts.

C++ designed Bjarne Stroustrup combined the ideas of 'C' and Simula. It added lots of interesting features without sacrificing efficiency.

Java was designed by a team consisting of James Gosling, Mike Sheridan and Patrick Naughton. Some of the main goals of Java are : support portability, make the language independent of computer systems(hardware, operating system), enable networking.
You may want to read this article.
Java White Paper
http://www.cs.dartmouth.edu/~mckeeman/cs118/references/OriginalJavaWhitepaper.pdf
Java has evolved over a period of two decades. The latest version of Java is java 1.8.

Language evolution never ends. There have been many more languages even

before Java. There are new languages designed every other day. But at the end of the day, Darwin's principles will rule. You can find which is the most popular language of the month and the statistics of usage of languages of that month in this link.

TIOBE: The intention of Being Earnest

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Object Oriented Programming Paradigm

There is a concept of type in the real world. Cars are of 4 wheeler automobile type. Mammals are a type of vertebrates. Students are of a type of homosapiens – may be a role that they play. The concept of type allows us to address all these entities uniformly – all students are interested in learning (hope so!), all politicians are .... (let me not say anything here lest it offends some people – I leave it to you to fill with whatever you think is suitable).

The concept of type indicates to us what can be a member of that type and what cannot be member of that type. A Civic car is a car – but a Sony TV is not a car. It also tells what an entity of that type can do and what it cannot. We can drive a car – but not a TV.

A type encapsulates a range(a set) of values and a set of operations.

We all know that integer is a type in Math. It says that 0 is an integer. For every integer, there is an integer whose value is one more than that of the integer in question. We can add two integers to give us an integer. We can multiply and so on.

What happens when we add 23 and 45? We should get 68 (unless we are into marketing or we assume that base is other than 10 – In the latter, the actual value will be equivalent to this result in that base!).

How did you add them? Add unit digits and propagate the carry? Add numbers row wise? Did you use fingers and toes? Did you use tally marks?

Math does not tell you how to add 23 and 45. It does tell you the expected result of addition. You are at liberty to choose the right method (what is right depends on you. In India, driving on the left of the road is right (Or should we say whatever is left of the road?). There is a unique way of implementing an operation.

A type tells 'what of operation' and not 'how of operation'

Programming languages provide a few types like integer, real (could be really complex – may support complex numbers of Math), boolean, character (not to be confused with those in dramas!). It can not support cars, clocks, fans, books, comics, movies unless it is developed for a particular domain.

So, language should provide us a mechanism to make our own type. Using those types we shooed be able to represent objects (or concepts) of the real world.
So, we get to the concept of a class.

Class is a mechanism to provide user defined types.

In a class, we provide the attributes – components that make up the object – A fan has blades, motor, on/off switch, speed regulator – A car has a steering wheel, Accelerator, Brake and Clutch levers, gear changer. In some languages, these are called data members or fields.

In a class, we also provide the methods – operations on these objects - start a fan, change speeds, stop the fan – start a car, accelerate, apply brakes, change gears. In some languages methods are also called functions or member functions.

We will just note that the class we write in many of these languages will also have implementation for the methods and therefore is not a pure type. In a pure type, we will only say **what** should be done and **not how.**

So far we have said that we can represent an entity of real world(Or imaginary – dreams, hallucinations, emotions, ...) into a class in the solution.

Let us consider an example of translating a real world concept into the solution domain. Let us ask ourselves a few question?

What is the time? (Should we say what is time? You may want to read My Date Less Diary of R K Narayan – He says that if somebody asks the time, you may be able to tell the time. If somebody asks what is time, it is very philosophical and therefore difficult to answer. He also says that if you have two watches, you do not have the time!). My computer says that the time now is 9:54 am. It represents one of measuring time – hours and minutes and am or pm. Should we use 12 hour clock or 24 hour clock? Can we represent time using the Indian system – ghalige and vighalige? Or are hours and minutes sufficient or should we also consider seconds ? What if I am a measuring time in a sprint? What if I am considering time with respect to the CPU speed? Should I consider nanoseconds? What about time across different countries? What about time across wide nations (East – West and not North – South)? Should the time also indicate day light saving? Should the time also have the date as part of it?

The point to note is that any object of real world will have various attributes. We select what is required for the problem we are trying to solve. There is unique representation for an entity of the real world. This concept is called abstraction.

We select what is essential. We ignore what is not essential. This selection depends on the observer.

If a topic is said to be optional in the syllabus or the teacher tells you that there will be no questions in the examination, you will abstract away those topics from your study scheme and your brains!

We create certain artifacts. For example, I may cook some food. (I used to make Bisibele bath – Once cooker exploded – now I avoid going near the stove – I am like Tenali Ramakrishan's Cat!). I will have to eat what I cook (In Kannada, we say 'Madid unno maharaya'). I want others to eat it too (guinea pigs!). So what I cook, if I am running a hotel, should depend what the users (or clients) want. We say in marketing that 'Clients are always right'. Our decisions of class design depend on what the clients want my class to do.

For example, in the case of time, I may want to display the time, change the time, find the difference between two times, check what will be the time after a time period. Based on these, we decide what the solution representation of the problem should have.

We put together these attributes and behaviour. This is called encapsulation.
We may also have access control in some languages.

We may decide that we will represent the time as hours (24 hr clock), and minted (0 – 59). We put them together so that the client can make an object of time together instead of creating hour and min separately. We put things together. In the good old days, each spice would be kept separately. Now we get spice boxes with partitions for different spices. So we can put all of them together. That makes searching for these spices easy.

Encapsulation is putting attributes and behaviour together.

Let us assume that we represent hours as hr and minutes as min in our solution domain. So the user can make an object of time and access hr and min and play with them. Life is happy here after.

But some client complains. What is min? Is that standing for minimum or minutes? Why not give better name? (Incidentally there was a fast bowler in England called Old. Even though he was the youngest player in the team first time he came to India, he was still Old!). So we decide to change the names to hour and minute. So nice. But the clients who are used hr and min will be in trouble. They can never access hr and min!

What is constant in this world is change.

Things change. We must be able to take care of changes.
We should put “no entry” board for the clients so that they will never get to know how we cook. (Have you even gone into the kitchen of a hotel? If you do, may be you will not be able to eat again!).

We want to hide what could change. We want to expose what will not change. We should guarantee to the client so that he will have little dependency on our design.

We use hiding concept to manage changes.

We hide what could change and we expose what shall not change.
We hide the implementation and expose the interface.

It is the normal convention to hide the attributes and expose some of the methods.
The way of passing message or invoking these exposed methods cannot be changed. If the interface is changed, all those depending on these will have to change their designs. This would have cascading effect.

Many think that the concept of encapsulation is security. The main idea of encapsulation is put attributes and behaviour together.
Also provide access control. Both these help in maintenance – that is the biggest challenge in software today!

An Object oriented language supports the following concepts:

- abstraction
- encapsulation
- composition
- inheritance
- polymorphism

We have discussed a couple of them here. We shall discuss the rest in the

following lectures.

Programming in Java

Computers even today are dumb – they understand their own language called the machine language in which we cannot easily converse. So, we write programs in languages which are far away from the computer – called high level languages. We use tool or tools to convert these instructions to whatever the computer can understand.

Java is one of such languages designed in the middle of 90s by a team led by Gosling. Programs in Java are converted to some intermediate form – called class files, There is another program called Java virtual machine (JVM) which understands these class files and then converts them to whatever the computer can respond. These class files do not depend on the computer and the operating system on which we are working. So, we can compile our Java programs on one computer, create the class files and take them to another computer with different hardware and operating system. That will also work as long as that computer also has Java virtual machine installed on it.

Java compiler has the name javac and Java virtual machine has the name Java.
These are also referred to as jdk(Java development kit) and jre(Java runtime environment) respectively.

As the programs in Java compile to an intermediate code called Java class files which are independent on the hardware and OS on which it works, we say that Java is architecturally neutral.

Program in Java follows the following structure.
1. The file name should have the extension .Java.
2. The file should have something called a class whose name is same as the filename without extension.
3. The program starts executing from a function(called method in Java) whose name is main. It takes array of Strings as arguments.
4. We use a method called print or println to display the output on the screen.

Filename : Example1.java

```
public class Example1

{

        public static  void main(String[] args)

        {

                System.out.println("hello world");

        }

}
```

We compile the program using the command javac. This generates a file called a class file. We can examine its content using a command called javap. We can generate some information about the program and its content(called documentation) using a command called javadoc.

Public indicates that the particular entity is an interface – it is accessible to the external world. In object world, we ask the object to do something for us. In the beginning there is no object. To break this shackle of chicken and egg, we make the main method a static method. This can be called without an object. We will discuss about these concepts at depth in the next lectures.