# Object Oriented Programming

## Learning outcomes:

On completion of this course unit, the student should be able to:

* describe the principles of object-oriented programming
* discuss the advantages of object-oriented approach
* design, develop, test and debug object oriented computer programs
* develop GUI using an object oriented programming language
* develop object oriented software applications with database connectivity  
    
  **When developing the object oriented application, it would be better if below listed aspects could be covered.  
    
  I would suggest to start an application at the very beginning and build it gradually by implementing the theoretical parts in to that application along the way.  
    
  Through such approach students will get a better understanding how the theory could be practically applied in Java.  
    
  By the end of lecture series they will have a fully functioning system with examples to refer in their hand when they build their own application for the assignment and practical test.  
    
  Its ok to use NetBeans IDE since GUI part needs to be built, but do not encourage to use convenient functions available in it like creating the DB connection from inside, adding external jars. The application should be runnable from outside even it is exported as a jar, and most importantly student should know things like what is the classpath, parameters in DB connection etc.**
* analyze a given scenario and build a solution using java,
  + identify the objects, their attributes and behaviors along with their communications with each other
  + use all data types available in Java, when mapping attributes of objects knowing when to use them, along with autoboxing in a meaningful way
  + use java collections (Lists/Maps/Sets), Enums when designing the java objects to map real world entities attributes
  + map those real world objects in to java code, use all OOP concepts in a way to demonstrate their advantages
    - Use Interfaces and Abstract classes - know similarities and differences
    - Use method overloading and overriding - know similarities and differences
    - Use associations - Association vs Aggregation vs Composition
    - Use encapsulation – know the advantages
  + Use LocalDate/ LocalDateTime (java.time) for date related attributes/methods opposed to java.util.Date  
    Use data formatting and parsing – java.util.Date conversions to java.time.LocalDate
  + Use StringBuffres/StringBuilders , Regex, and functionality available in String class itself
  + use Generics, static methods, Exceptions when mapping real world object behaviors and communications in to java
  + Use for loops, enhanced for loops. Lambda expressions, if else statements, switch case statements
  + design the relational database structure, to store necessary details (to develop ORM skills)
  + Java JDBC - Know parameters of a DB connection, preparedstamenets, resultsets
  + use threads when and where applicable - eg: - async tasks/scheduled tasks – know ways to create a thread
  + use meaningful packages/ class names/ variable names/ method names – follow conventions (camel case) and rules
  + structure the code properly using packages (controller layer (GUIs in swing)/ service layer/domain layer/data access layer/util layer)
  + File manipulation – use File/FileInputStream/FileOutputStream/BufferedReader/ BufferedWriter/FileWriter/FileReader

* + On bit advanced topics
    - Try out annotations – use existing simple ones like @Override and try to create own annotations and use in code
    - Try out reflection – modify java objects in runtime, note the usages
    - Know about external dependencies (jar files), to use more functionality than available in standards JDK – how to include external jars, how to use
    - Write a small utility application (eg: to add two numbers), export as a jar and use in another application
    - java.nio package – socket related programming
    - Java EE – servlet/jsp, war files, JEE frameworks, ORM tools
    - Java ME – Android
    - Eclipse IDE

## Course content:

### Principles of Object Oriented Programming (OOP),

##### Introduction to Java

###### What is Java

* Java is a high level, robust, secured and object-oriented programming language and computing platform
* Java is fast, secure, and reliable.

###### History of Java

* + James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991.
  + The small team of sun engineers called Green Team.
  + Originally designed for small, embedded systems in electronic appliances like set-top boxes.
  + According to James Gosling the name “Java” was one of the top choices along with “Silk".
  + Since java was so unique, most of the team members preferred java.
  + Java is an island of Indonesia where first coffee was produced (called java coffee).
  + “Java” is just a name not an acronym.
  + Java was originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995.
  + JDK 1.0 was released in (January 23, 1996).
  + Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Java Version | Code Name | Release Date | Main Features Introduced |
| JDK Version 1.0 | Oak | January 23, 1996 | Initial Release to public |
| JDK Version 1.1 | - | February 19, 1997 | * JDBC (Java Database Connectivity) * Inner Classes * Java Beans * RMI (Remote Method Invocation) * Reflection (introspection only) |
| J2SE Version 1.2 | Playground | December 8, 1998 | * Collections framework. * Java String memory map for constants. * Just In Time (JIT) compiler. * Jar Signer for signing Java ARchive (JAR) files. * Policy Tool for granting access to system resources. * Java Foundation Classes (JFC) which consists of Swing 1.0, Drag and Drop, and Java 2D class libraries. * Java Plug-in * Scrollable result sets, BLOB, CLOB, batch update, user-defined types in JDBC. * Audio support in Applets. |
| J2SE Version 1.3 | Kestrel | May 8, 2000 | * Java Sound * Jar Indexing * A huge list of enhancements in almost all the java area. |
| J2SE Version 1.4 | Merlin | February 6, 2002 | * XML Processing * Java Print Service * Logging API * Java Web Start * JDBC 3.0 API * Assertions * Preferences API * Chained Exception * IPv6 Support * Regular Expressions * Image I/O API |
| J2SE Version 5.0 | Tiger | September 30, 2004 | * Generics * Enhanced for Loop * Autoboxing/Unboxing * Typesafe Enums * Varargs * Assertions * Static Import * Metadata (Annotations) * Instrumentation |
| Java Version SE 6 | Mustang | December 11, 2006 | * Scripting Language Support * JDBC 4.0 API * Java Compiler API * Pluggable Annotations * Native PKI, Java GSS, Kerberos and LDAP support. * Integrated Web Services. * Lot more enhancements. |
| Java Version SE 7 | Dolphin | July 28, 2011 | * Strings in switch Statement * Type Inference for Generic Instance Creation * Multiple Exception Handling * Support for Dynamic Languages * Try with Resources. * Java nio Package * Binary Literals, underscore in literals. * Diamond Syntax * Automatic null Handling |
| Java Version SE 8 | culture | March 18th 2014 | * Lambda Expressions * Pipelines and Streams * Date and Time API * Default Methods * Type Annotations * Java nio Package * Concurrent Accumulators * Parallel operations * PermGen Error Removed * TLS SNI |

###### Main Types of Java Applications

* + Standalone Application
    - Known as desktop application or window-based application.
    - Needs to be installed on the machine to be used. E.g.:- an image editor, media player
    - AWT and Swing are used in java for creating standalone applications.
  + Web Application
    - An application that runs on the server side and creates dynamic web pages/ content, is called web application.
    - These can be accessed from client side usually through a browser
    - Servlet/JSP, Struts, Spring, JSF like technologies are used for creating web applications in java.
  + Enterprise Application
    - An application that is distributed in nature, such as banking applications.
    - It has the characteristics like high level security, load balancing and clustering.
    - In java, EJB is used for creating enterprise applications.
  + Mobile Application
    - An application that is created for mobile devices.
    - Currently Android and Java ME are used for creating mobile applications.
  + Embedded Systems
    - Very lightweight programs to run inside a limited resources

###### Where it is used

* + From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere
  + Android Applications
    - Andorid applications are written Java programming language, with Google's Android API, which is similar to JDK
    - Android uses different JVM and different packaging, but code is still written in Java
  + Server Apps at Financial Services Industry
    - Vastly used in Financial Services due to secured nature
    - Java is mostly used to write server side application, mostly without any front end, which receives data form one server (upstream), process it and sends it other process (downstream).
  + Java Web applications
    - Java is popular on E commerce and web application space.
    - A lot of RESTful services being created using JEE frameworks such as Spring MVC, Struts 2.0 and similar frameworks.
    - Simple Servlet, JSP and Struts based web applications are quite popular even today and still in use
    - Many of government, transport, education, commercial and several other departments have their web application built in Java.
  + Software Tools
    - Many useful software and development tools are written and developed in Java
      * e.g. Eclipse, InetelliJ Idea and Netbans IDE – most used desktop applications written in Java
  + Embedded Software
    - Embedded systems, ranging from tiny chips to specialized computers, are components of larger electromechanical systems performing dedicated tasks.
    - Platform independence and easiness to produce bug-free software in Java than in C or C++, make it widely used for embedded systems.
    - Java has been designed from the ground up to produce code that is simpler to write and easier to maintain.
    - So many devices, such as SIM cards, blue-ray disk players, utility meters and televisions, use embedded Java technologies.

* + Big Data technologies
    - Even though Java is not dominating in this category, Hadoop and other big data technologies are also using Java in certain parts
    - e.g. Apache's Java-based HBase and Accumulo (open source), and ElasticSearch as well.
  + Scientific Applications
    - Java is the choice of many software developers for writing applications involving scientific calculations and mathematical operations.
    - These programs are generally considered to be fast and secure, have a higher degree of portability and low maintenance.
    - Applications like MATLAB use Java both for interacting user interface and as part of the core system.

###### Java terminology

* + Java Development Kit (JDK)
    - This is complete java development kit that includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc.
    - JDK should be installed in the machine to create, compile and run Java programs.
  + Java Runtime Environment (JRE)
    - JRE is a part of JDK which means that JDK includes JRE.
    - JRE can be installed without JDK, a java program could be run only with JRE.
    - Java program can’t be compiled with JRE, for that, the JDK is required.
    - JRE includes JVM, browser plugins and applets support.
    - When only need to run a java program on the computer, only JRE.
  + Download and Install JDK 8 (1.8) in the machine
    - Set JAVA\_HOME variable in OS
      * Eg:- C:\Program Files\Java\jdk1.8.0\_60
    - Add JAVA\_HOME/bin to PATH variable in OS
      * Eg:- C:\Program Files\Java\jdk1.8.0\_60\bin
  + Java Virtual Machine (JVM)
* The Java Virtual Machine is a program, for a particular hardware and software platform, that runs Java applications.
* Primary function of JVM is to execute the bytecode produced by compiler.
* Each operating system has different JVM, however the output they produce after execution of bytecode is same across all operating systems.
* That is why we call java as platform independent language.
  + bytecode
    - javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM.
  + The phases of program execution in Java
    - write the program in Java
      * By a programmer
      * Saved as a .java file – Student.java
    - compile the program using Java Compiler
      * javac is the primary java compiler included in java development kit (JDK).
      * It takes java program as input and generates java bytecode as output
      * Saved as a .class file – Student.class
    - run the program – using JVM
      * JVM executes the bytecode generated by compiler.
      * This is called program run phase
    - The bytecode is saved in a .class file by compiler
  + Write a simple Java class, compile it and run using command line



* Compile :- D:\>javac HelloWorld.java
* Run :- D:\>java HelloWorld
  + - Output should be :- Hello World!

###### Java Programming Language Platforms

* + All Java platforms consist of a Java Virtual Machine (VM) and an application programming interface (API).
  + Each Java platform provides a virtual machine and an API, and this allows applications written for that platform to run on any compatible system with all the advantages of the Java programming language: platform-independence, power, stability, ease-of-development, and security.
  + An API is a collection of software components that can be used to create other software components or applications.
  + There are four platforms of the Java programming language:
    - Java Platform, Standard Edition (Java SE)
      * Java SE's API provides the core functionality of the Java programming language.
      * It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.
      * In addition to the core API, the Java SE platform consists of a virtual machine, development tools, deployment technologies, and other class libraries and toolkits commonly used in Java technology applications.
    - Java Platform, Enterprise Edition (Java EE)
      * The Java EE platform is built on top of the Java SE platform.
      * The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications
    - Java Platform, Micro Edition (Java ME)
      * The Java ME platform provides an API and a small-footprint virtual machine for running Java programming language applications on small devices, like mobile phones.
      * The API is a subset of the Java SE API, along with special class libraries useful for small device application development.
      * Java ME applications are often clients of Java EE platform services.
* JavaFX
  + JavaFX is a platform for creating rich internet applications using a lightweight user-interface API.
  + JavaFX applications use hardware-accelerated graphics and media engines to take advantage of higher-performance clients and a modern look-and-feel as well as high-level APIs for connecting to networked data sources.
  + JavaFX applications may be clients of Java EE platform services.

###### Java Features

* + Simple
    - According to Sun, Java language is simple because:
      * Syntax is based on C++ (so easier for programmers to learn it after C++).
      * Removed many confusing and/or rarely-used features
        + e.g., explicit pointers, operator overloading etc.
      * No need to remove unreferenced objects because there is Automatic Garbage Collection in java.
  + Object-Oriented
    - Means software is organized as a combination of different types of objects that incorporates both data and behavior.
    - Object-oriented programming (OOPs) is a methodology that simplify software development and maintenance by providing some rules.
    - Basic concepts of OOPs are:
      * Object
      * Class
      * Inheritance
      * Polymorphism
      * Abstraction
      * Encapsulation
  + Platform independent
    - A platform is the hardware or software environment in which a program runs.
    - There are two types of platforms
      * software-based
      * hardware-based.
    - Java provides software-based platform.
    - The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:
      * Runtime Environment
      * API(Application Programming Interface)
    - Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, and Mac/OS etc.
    - Java code is compiled by the compiler and converted into bytecode.
    - This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere (WORA).



* + Portable
    - The generated bytecode may be carried to any platform and executed if a JRE is available at the destination
      * Eg: a jar file generated in a Windows machine could be transferred to an Ubuntu machine and executed without any issue.
  + Secured
    - Java is secured because
      * No explicit pointer
      * Java Programs run inside virtual machine sandbox
    - Classloader:
      * Adds security by separating the package for the classes of the local file system from those that are imported from network sources.
    - Bytecode Verifier:
      * Checks the code fragments for illegal code that can violate access right to objects.
    - Security Manager:
      * Determines what resources a class can access such as reading and writing to the local disk.
    - These security are provided by java language.
    - Some security can also be provided by application developer through SSL, JAAS, and Cryptography etc.



* + Robust
    - Robust means strong.
      * Java uses strong memory management.
      * Lack of pointers that avoids security problem.
      * Automatic garbage collection in java.
      * Exception handling and type checking mechanism in java.
  + Architecture neutral
    - There is no implementation dependent features
      * e.g. size of primitive types is fixed.
    - In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture.
    - But in java, it occupies 4 bytes of memory for both 32 and 64 bit architectures.
  + High Performance
    - Java is faster than traditional interpretation since byte code is "close" to native code still somewhat slower than a compiled language (e.g., C++)
  + Multithreaded
    - A thread is like a separate program, executing concurrently.
    - Java programs could be written that deal with many tasks at once by defining multiple threads.
    - The main advantage of multi-threading is that it doesn't occupy memory for each thread.
    - It shares a common memory area.
    - Threads are important for multi-media, Web applications etc.
  + Distributed
    - Distributed applications can be created in java.
    - RMI and EJB are used for creating distributed applications.
    - Allows to access files by calling the methods from any machine on the internet.

#### Classes and objects

#### Information hiding and encapsulation

#### Inheritance

#### Polymorphism, static and dynamic binding

#### Interfaces and Abstract classes

#### Exception Handling

#### GUI Components

#### Generic Classes and Methods

#### Accessing Databases with JDBC

#### Multithreading

# References

(n.d.). Retrieved from Java tutorial for beginners with examples: https://beginnersbook.com/java-tutorial-for-beginners-with-examples/

*Code Names for Java Versions*. (n.d.). Retrieved from Java2Learn: http://java2learn.com/code-names-for-java-versions/

Jaiswal, S. (n.d.). *java-tutorial*. Retrieved from javatpoint.com: https://www.javatpoint.com/java-tutorial

*What is Java technology and why do I need it*. (n.d.). Retrieved from java.com: https://www.java.com

*Where is Java used in Real World*. (2014, 12 03). Retrieved from javarevisited.blogspot.com: http://javarevisited.blogspot.com/2014/12/where-does-java-used-in-real-world.html