Classpath Vs Path

classpath:: describe a location where required class are available This is used by JVM .If classpath is missed then our project may be not compile or may not be run.

javac Test.java -> this javac line is responsible to validate the whole program .This internally use javac.exe file (binary executable file) .

path:: this describe the location where javac.exe file are present. (Javac.exe ->available in side the JDK\bin)

Issue With Old Date and Time classes before Java 8

There have been several problems with the date and time related classes before java 8, some of them are:

- Java Date Time classes are not defined consistently, we have Date Class in both **java.util** as well as **java.sql** packages. Again formatting and parsing classes are defined in **java.text** package.
- **java.util.Date** contains both date and time, whereas **java.sql.Date** contains only date. Having this in **java.sql** package doesn't make sense. Also both the classes have same name, that is a very bad design itself.
- All the Date classes are mutable, so they are not thread safe. It's one of the biggest problem.
- Date class doesn't provide internationalization, there is no timezone support. So **java.util.Calendar** and **java.util.TimeZone** classes were introduced, but they also have all the problems listed above.

Java 8 Date/Time API

Java 8 Date Time API is designed to overcome all the flaws in the legacy date time implementations. Some of the design principles of new Date Time API are:

- **Immutability:** All the classes in the new Date Time API are immutable and good for multithreaded environments.
- **Separation of Concerns:** The new API separates clearly between human readable date time and machine time (unix timestamp). It defines separate classes for date, time, Datetime, Timestamp, Timezone etc.
- **Clarity:** The methods are clearly defined and perform the same action in all the classes. For example, to get the current instance we have now() method. There are format() and parse() methods defined in all these classes rather than having a separate class for them.
  - All the classes use Factory Pattern and Strategy Pattern for better handling. Once you have used the methods in one of the class, working with other classes won't be hard.
- **Utility operations:** All the new Date Time API classes comes with methods to perform common tasks, such as plus, minus, format, parsing, getting separate part in date/time etc.

Java 8 Date Time API consists of following packages

- java.time
- java.time.chrono
- java.time.format
- java.time.temporal
- java.time.zone

1. **java.time :** This is the base package of new Java Date/Time API. All the major base classes are part of this package, such as LocalDate, LocalTime, LocalDateTime, Instant, Period, Duration etc. All of these classes are immutable and thread safe. Most of the times, these classes will be sufficient for handling common requirements.
2. **java.time.chrono :** This package defines generic APIs for non ISO calendar systems. We can extend AbstractChronology class to create our own calendar system.
3. **java.time.format:** This package contains classes used for formatting and parsing date time objects. Most of the times, we would not be directly using them because principle classes in java.time package provide formatting and parsing methods.
4. **java.time.temporal :** This package contains temporal objects and we can use it for find out specific date or time related to date/time object. **For example**, we can use these to find out the first or last day of the month.
5. **java.time.zone :** This package contains classes for supporting different time zones and their rules.

After Java 8

	Interface	Abstract Class
Constructors	✗	✓
Static Fields	✓	✓
Non-static Fields	✗	✓
Final Fields	✓	✓
Non-final Fields	✗	✓
Private Fields & Methods	✗	✓
Protected Fields & Methods	✗	✓
Public Fields & Methods	✓	✓
Abstract methods	✓	✓
Static Methods	✓	✓
Final Methods	✗	✓
Non-final Methods	✓	✓
Default Methods	✓	✗