LOVELY PROFESSIONAL UNIVERSITY

INTERNSHIP ON JAVA

SUBMITTED BY : SIMRAN MANHAS

SECTION : K18FR

REGISTRATION NO. : 11803523

CONTENTS

→ Classes and objects

→ Collections - 1

→ Relationships

→ Inheritance and polymorphism

→ Abstract and interface

→ Collctions-2

→ String/ Stringbuffer /String Tokenizer

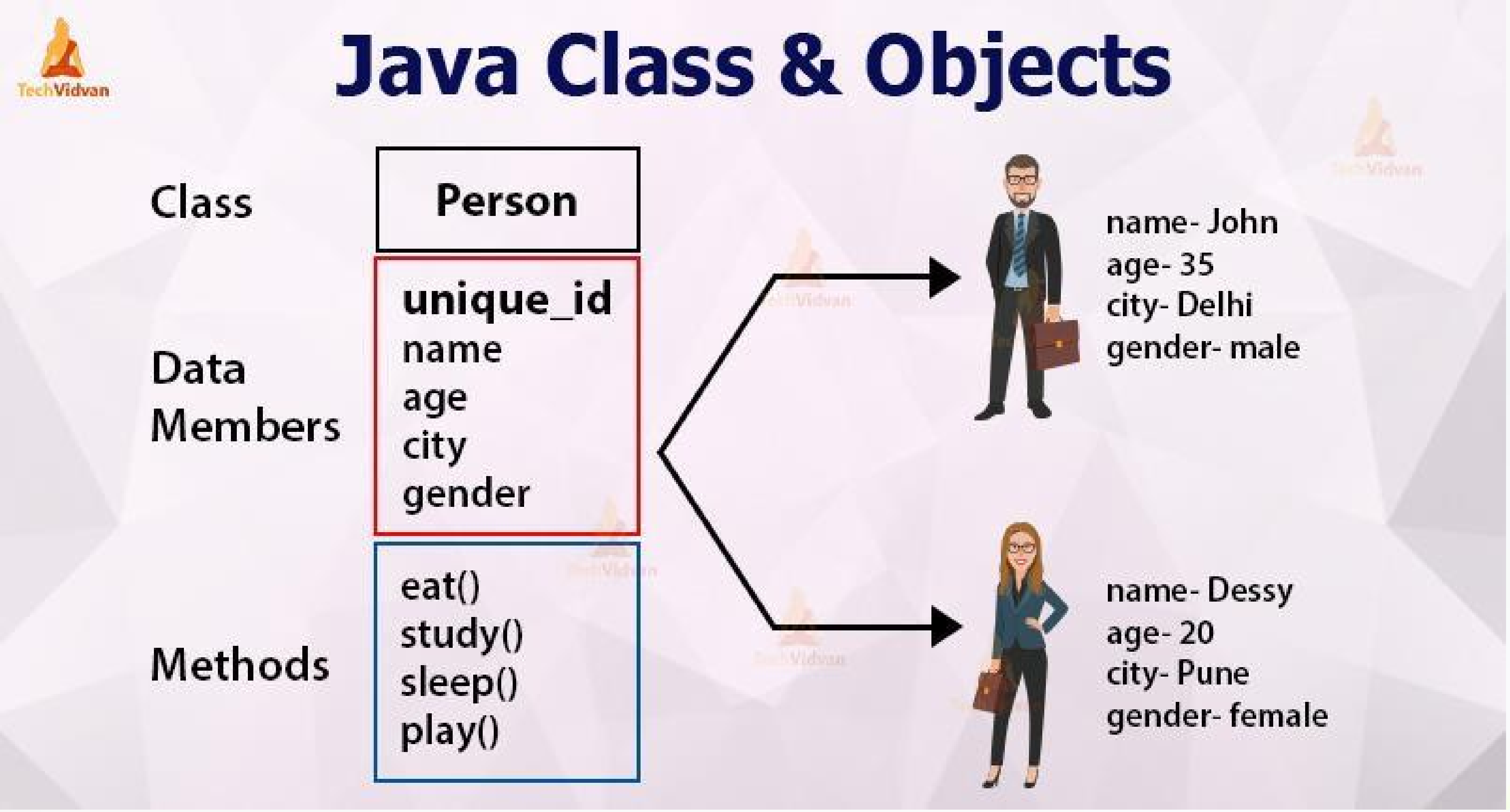
→ Exception handling

→ JDBC

→ Multithreading

→ Streams and Writers

→ Application Development



Java Class & Objects

Class	Person
Data Members	**unique_id** name age city gender

Methods	eat()
	study()
	sleep()
	play()

name- John
age- 35
city- Delhi
gender- male

name- Dessy
age- 20
city- Pune
gender- female

Classes and Objects

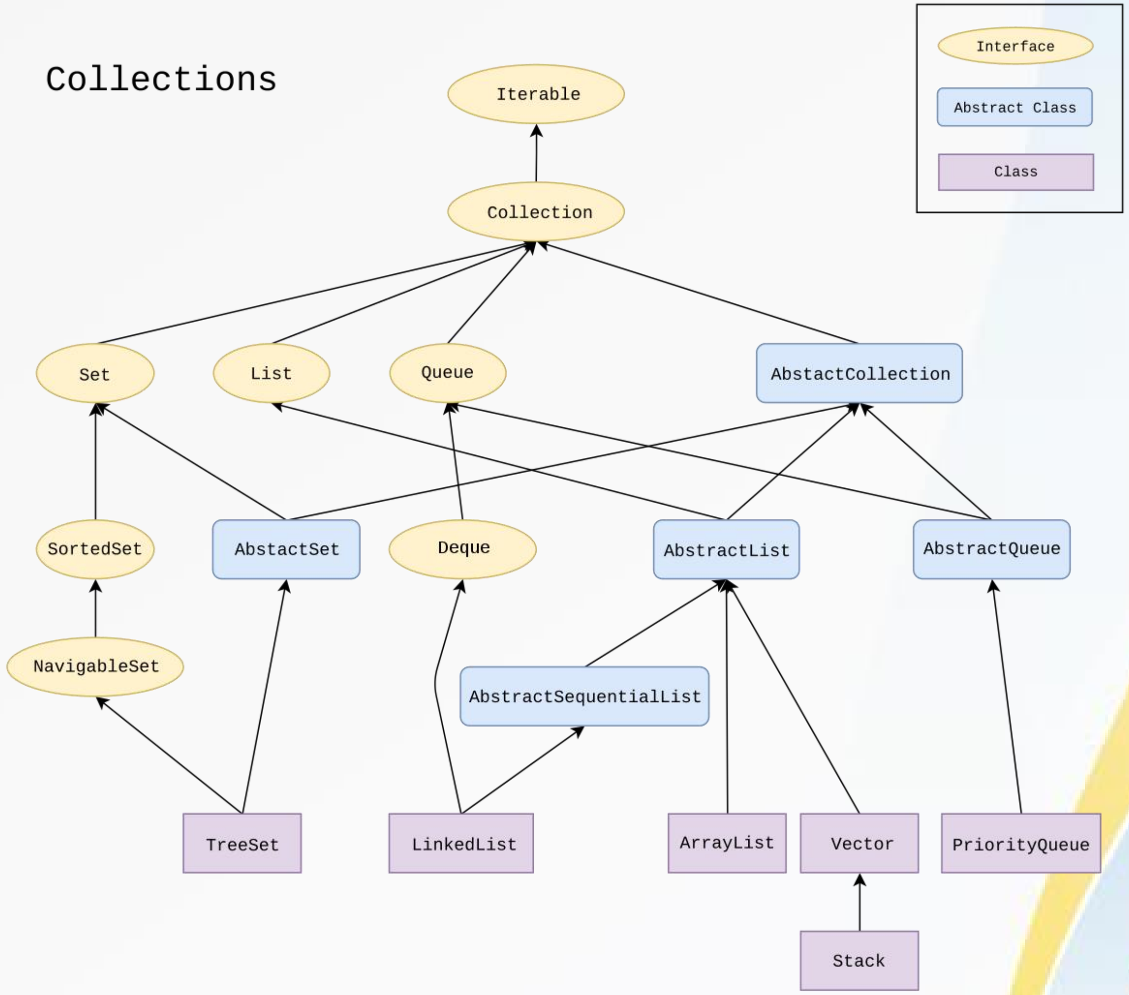
- **Java** is an **object-oriented programming language.**
- Everything in **Java** is associated with **classes and objects**, along with its attributes and methods.
- For example: In real life, a car is an **object**. ...
  A **Class** is like an **object** constructor, or a "blueprint" for creating **objects**.

COLLECTIONS

The **Java collections** framework is a set of classes and interfaces that implement commonly reusable **collection** data structures.

Although referred to as a framework, it works in a manner of a library.

The **collections** framework provides both interfaces that define various **collections** and classes that implement them.

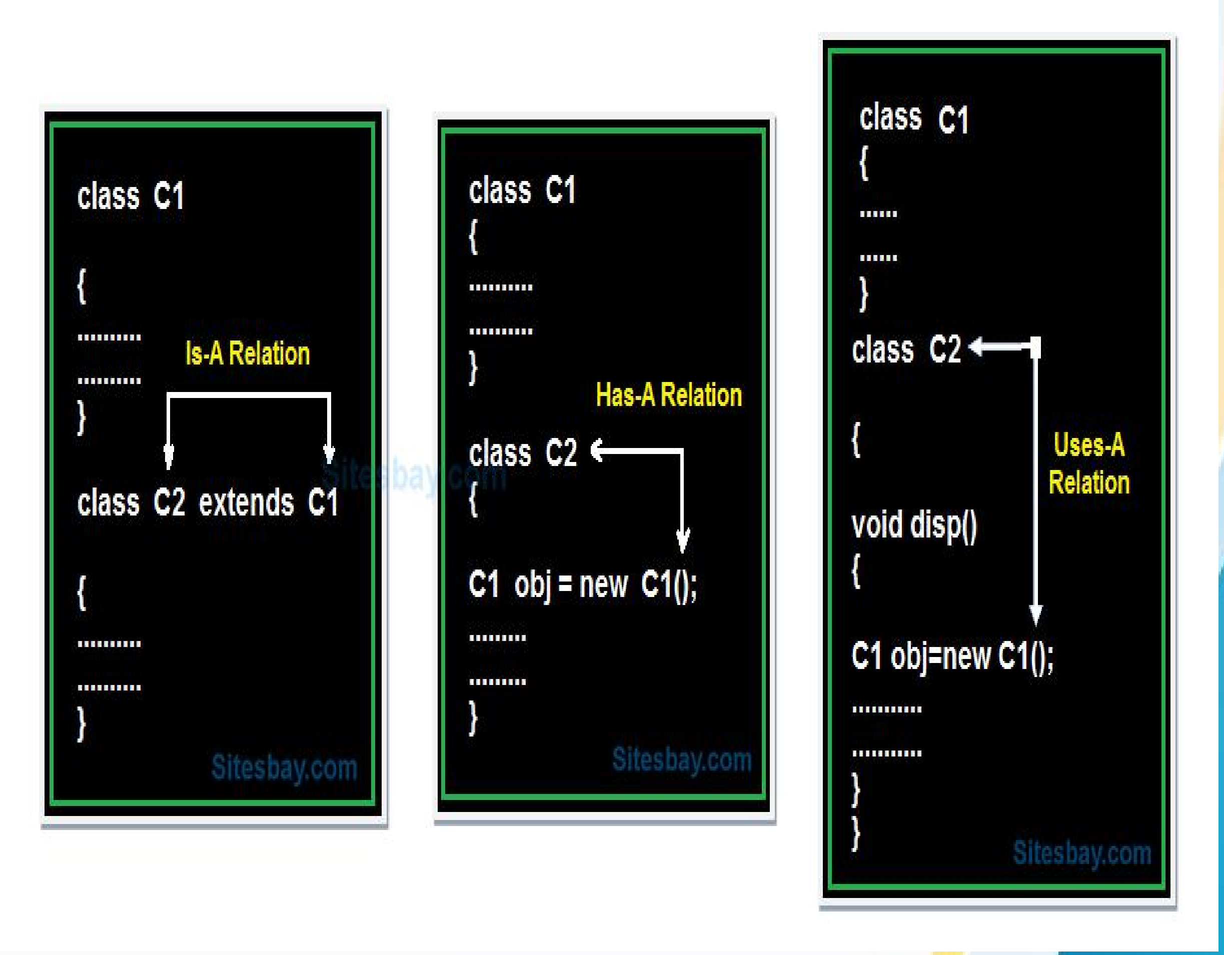


RELATIONSHIPS

A **Relationship** in **JAVA**

Has a **relationship** in **Java** is known to be as Composition. It is used for code reusability.

Basically, it **means** that an instance of the one class has a reference to the instance of another class or the other instance of the same class.



```
class C1

{
..........
..........
}

class C2 extends C1

{
..........
..........
}
```

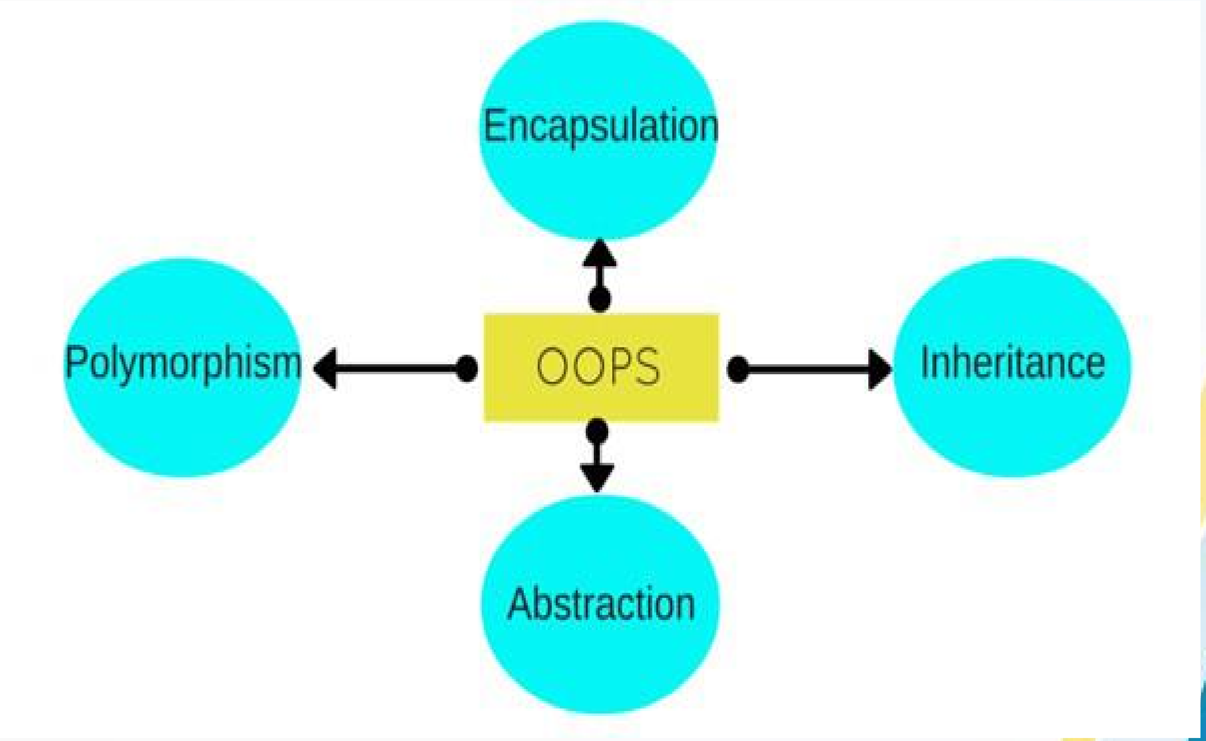
```
class C1
{
..........
..........
}

class C2
{

C1 obj = new C1();
.........
.........
}
```

```
class C1
{
......
......
}
class C2

{

void disp()
{

C1 obj=new C1();
..........
..........
}
}
```

INHERITANCE AND POLYMORPHISM

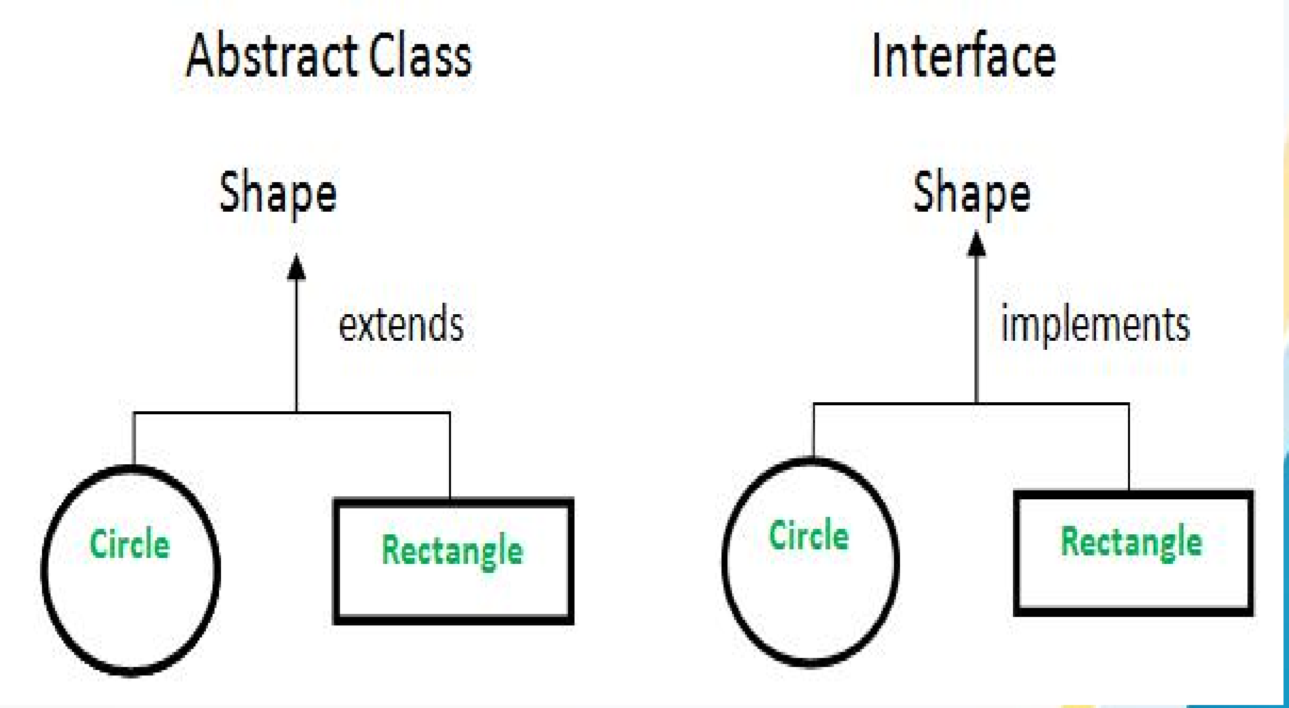
→**Polymorphism** means "many forms", and it occurs when we have many classes that are related to each other by **inheritance**.

→ **Inheritance** lets us **inherit** attributes and methods from another class. **Polymorphism** uses those methods to perform different tasks.



ABSTRACT AND INTERFACE

Main difference is methods of a **Java interface** are implicitly **abstract** and cannot have implementations. Variables declared in a **Java interface** is by default final. An **abstract** class may contain non-final variables. Members of a **Java interface** are public by default.



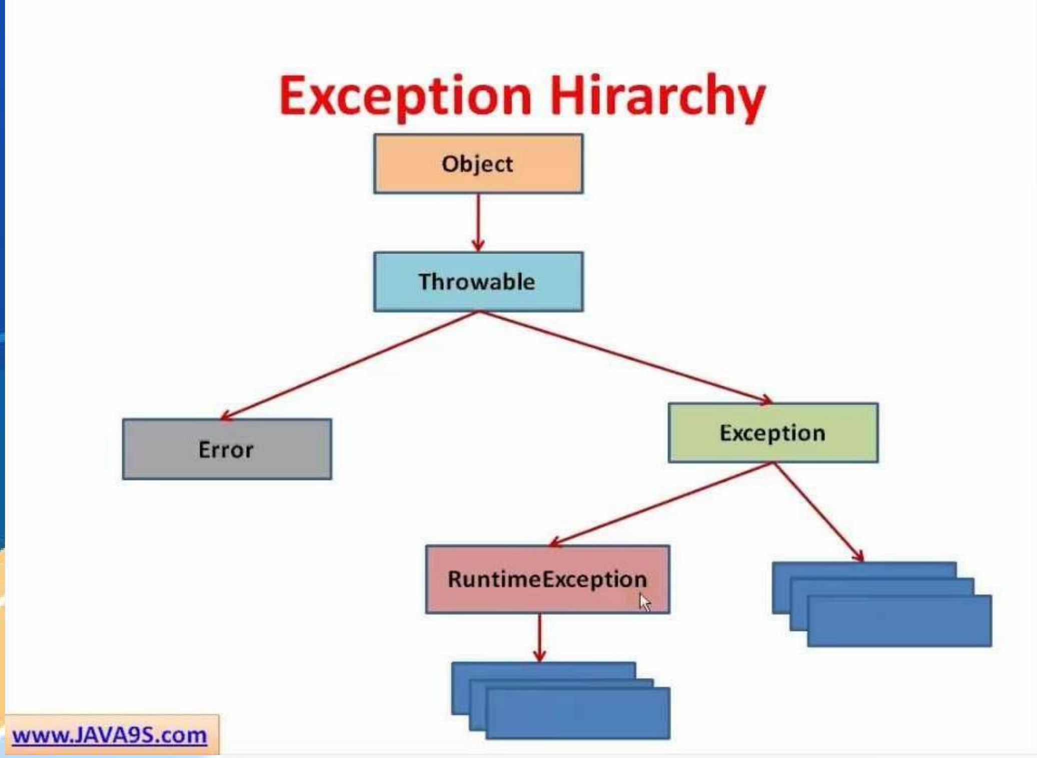
STRING / STRING BUFFER/ STRING TOKENIZER

A **string** is a data type used in programming, such as an integer and floating point unit, but is used to represent text rather than numbers. It is comprised of a set of characters that can also contain spaces and numbers.

A **string buffer** is like a String, but can be modified. It contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls. They are safe for use by multiple threads. Every string buffer has a capacity.

The **string tokenizer** class allows an application to break a string into tokens. The tokenization method is much simpler than the one used by the StreamTokenizer class. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments.

EXCEPTIONAL HANDLING



The **Exception Handling in Java** is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

Here, we will learn about Java exceptions, its type and the difference between checked and unchecked exceptions.

JDBC

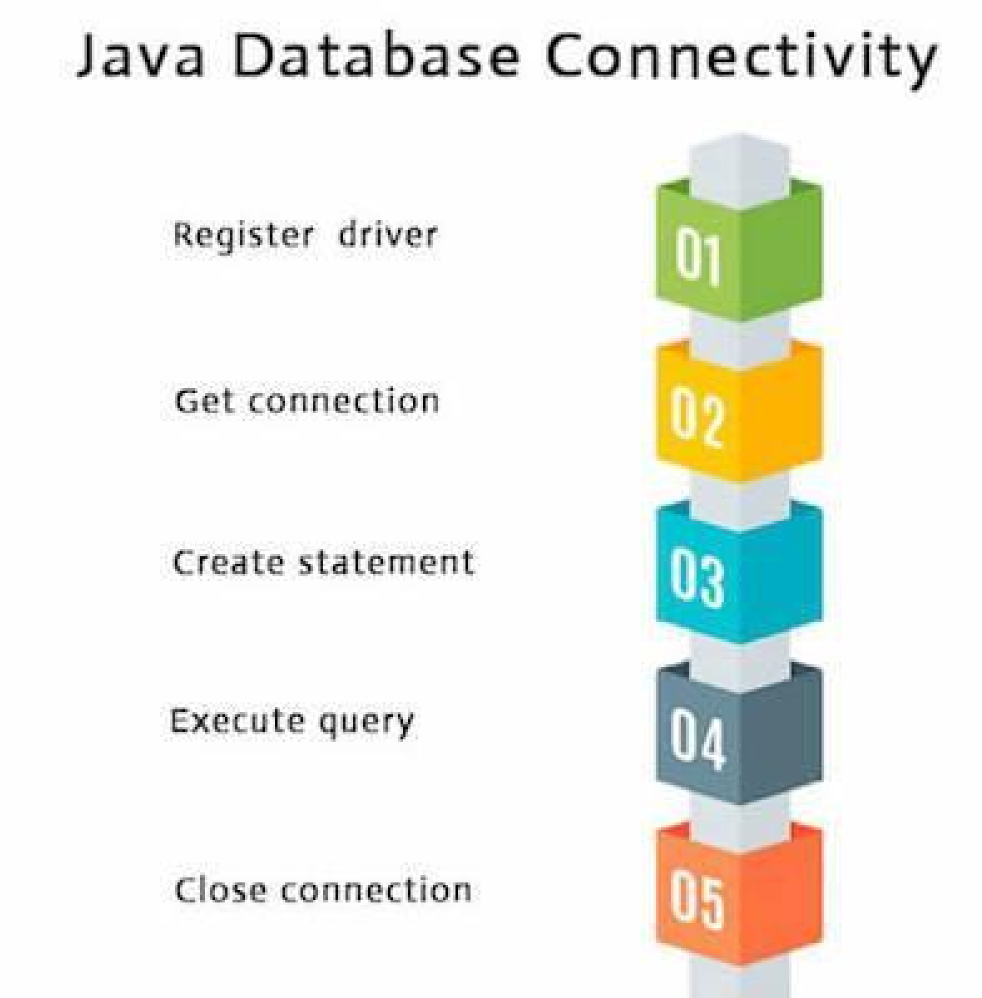
JDBC stands for **Java Database Connectivity.** JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database.

There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



MULTITHREADING

Multithreading in JAVA is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

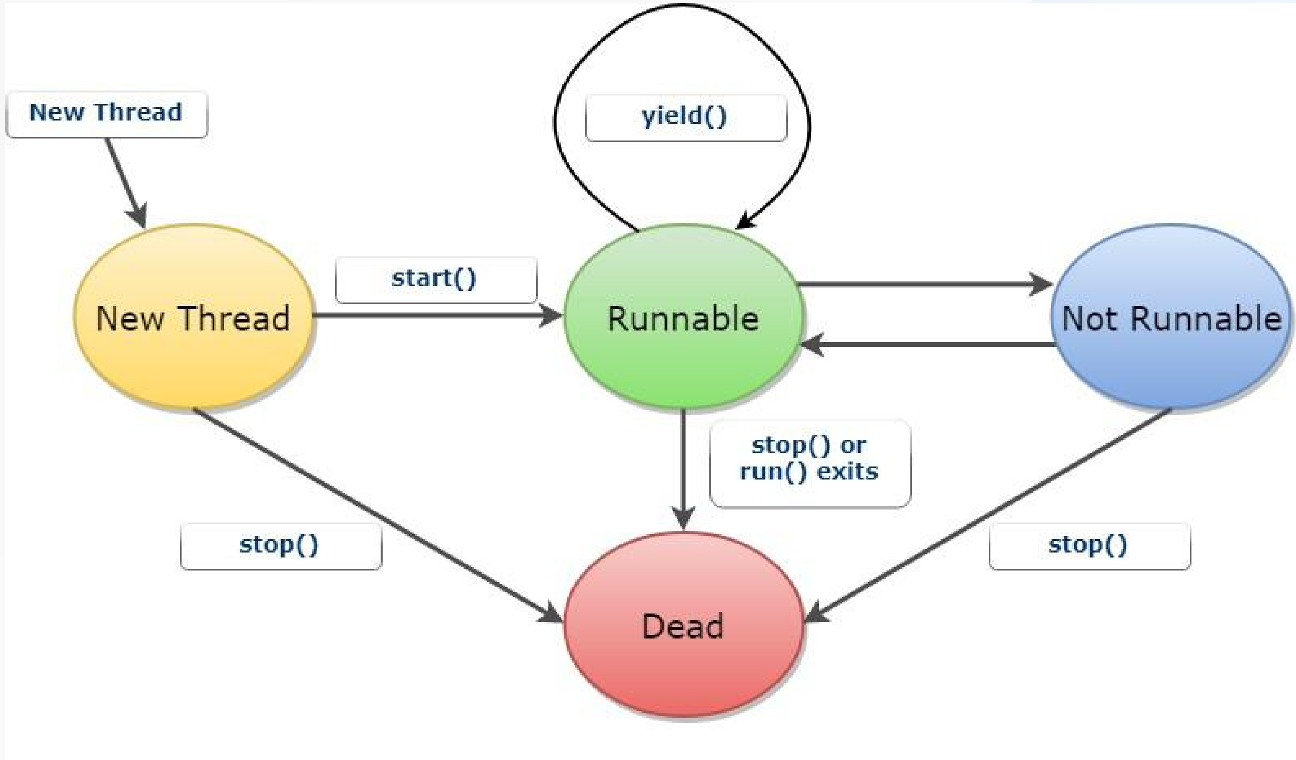
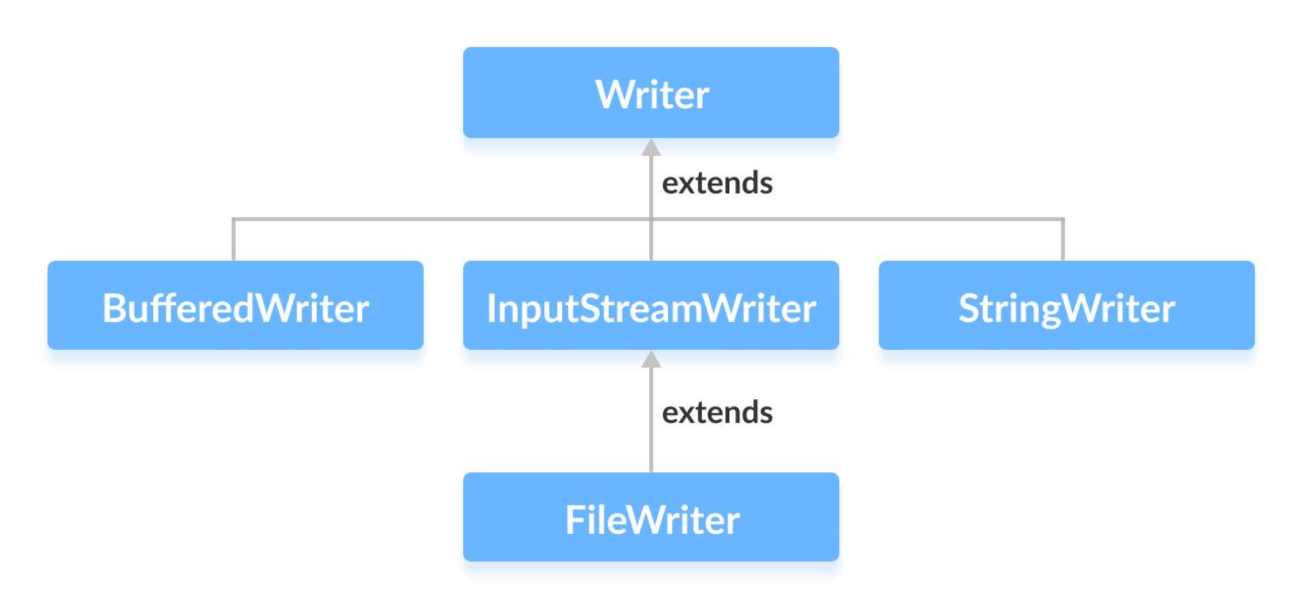


Fig: Life Cycle of a Thread in Java

STREAMS AND WRITER

Reading and Writing Files. As described earlier, a **stream** can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination. Here is a hierarchy of classes to deal with Input and Output **streams**.



APPLICATION DEVELOPMENT

Developing General Java Applications

•Project Setup. Creating a **Java** Class Library Project. ...

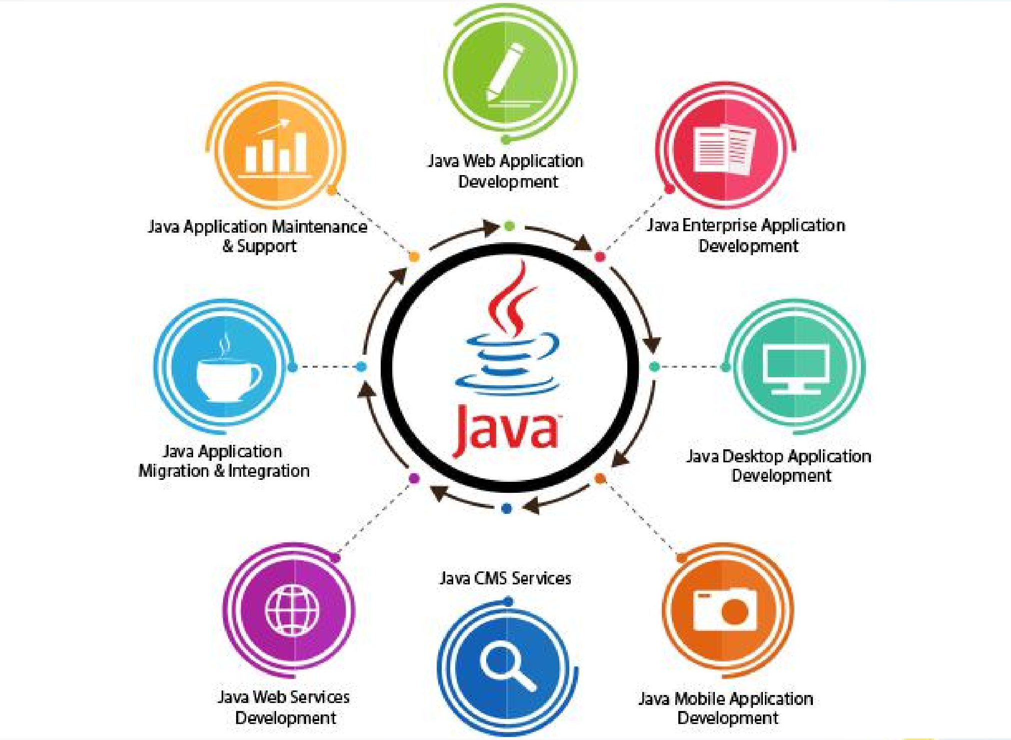
•Creating and Editing **Java** Source Code. Creating a **Java** Package and Class File. ...

•Compiling and Running the **Application**. ...

•Testing and Debugging the **Application**. ...

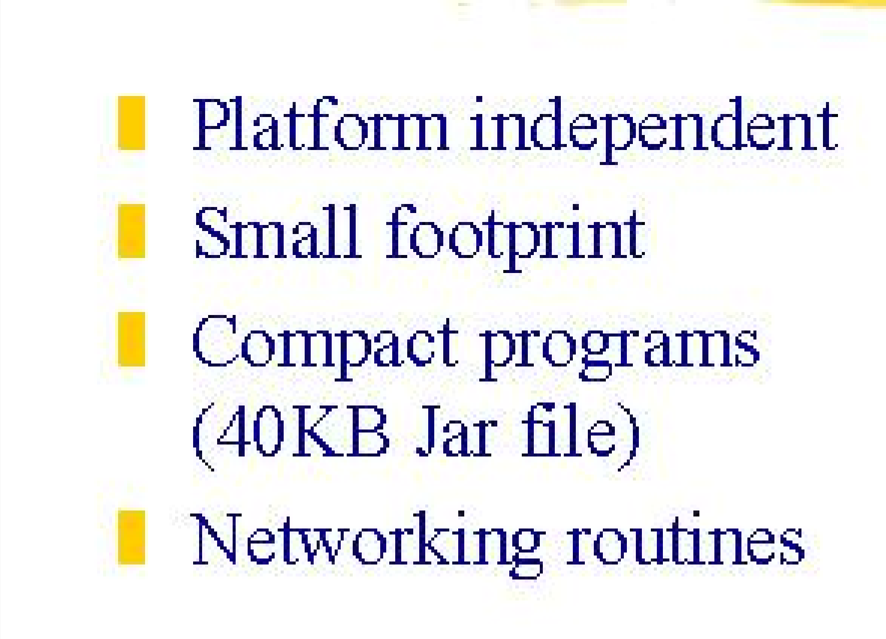
•Building, Running, and Distributing the **Application**. ...

•Other Common Tasks.


Java Web Application Development
Java Application Maintenance & Support
Java Enterprise Application Development
Java
Java Application Migration & Integration
Java Desktop Application Development
Java CMS Services
Java Web Services Development
Java Mobile Application Development

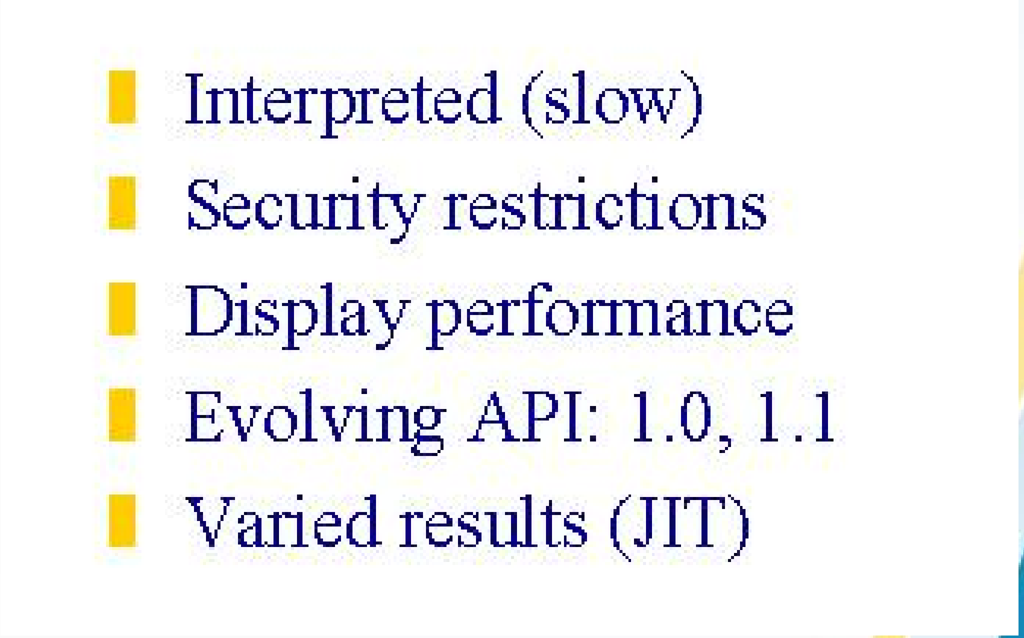
JAVA

ADVANTAGES



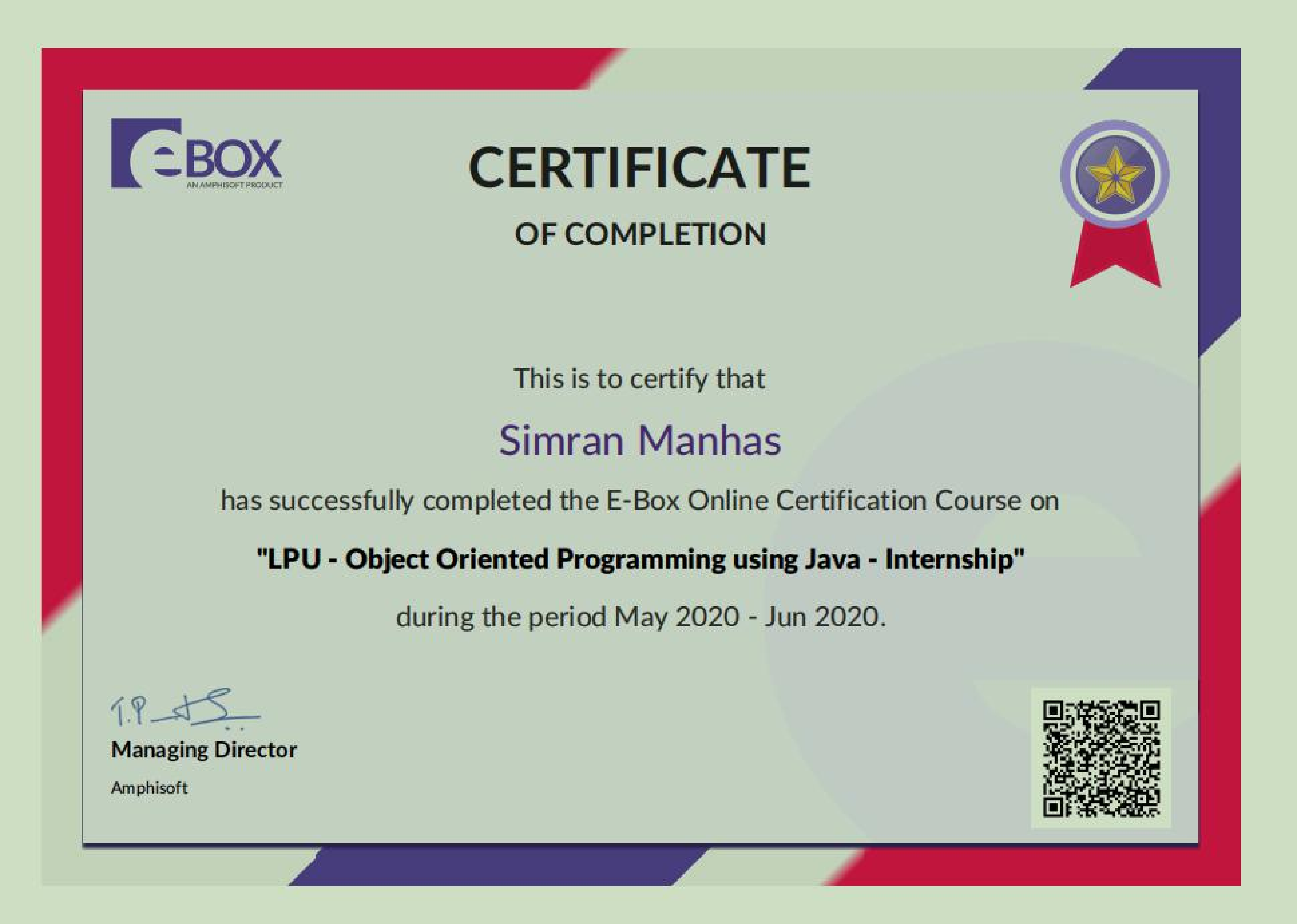
- Platform independent
- Small footprint
- Compact programs (40KB Jar file)
- Networking routines

DISADVANTAGES



- Interpreted (slow)
- Security restrictions
- Display performance
- Evolving API: 1.0, 1.1
- Varied results (JIT)

CERTIFICATION



eBOX
AN AMPHISOFT PRODUCT

CERTIFICATE

OF COMPLETION

This is to certify that

Simran Manhas

has successfully completed the E-Box Online Certification Course on

"LPU - Object Oriented Programming using Java - Internship"

during the period May 2020 - Jun 2020.

Managing Director
Amphisoft

THANKYOU