**Persistence**

Persistence is the characteristic where state (stored information at any given time) outlives the life of the process (instance of a computer program that is being executed) that created it. This is usually achieved by storing the state as data in non-volatile storage such as a hard drive or flash storage.

Examples of Persistence techniques:

* System Images – System Images is a copy of the entire state of a computer system.
* Journals – Journaling is the process of storing events in a log before each one is applied to a system.
* Dirty Writes – Dirty writes is the writing to storage of only those portions of system that have been modified since their last write.

**JDBC vs ORM**

JDBC – Java Database Connectivity is an API for Java programming language that defines how a client may access a database.

ORM – Object Relational Mapping is a programming technique for converting data between incompatible systems in object-orientated programming languages.

Examples that make use of JDBC or ORM: Hibernate, JPA, Spring etc.

**Reflection**

Reflection allows inspection of classes, interfaces, fields and methods at runtime without knowing the names of the interfaces, fields, or methods at compile time. It also allows instantiation of new objects and invocation of methods.

Examples of Reflection in Java are:

* java.awt – Abstract Window Toolkit provides a basic set of GUI widgets
* javax.swing – Swing is a collection of routines that build on java.awt to provide a platform independent widget toolkit.
* java.beans – Various classes for developing and manipulating beans (reusable components defined by Java Beans)
* java.rmi – provides Java remote method invocation to support remote procedure calls, between two java applications running in different virtual machines
* java.sql - An implementation of the JDBC IPI is grouped into java.sql

**Remote Invocation**

Remote Invocation…

Request-Reply Protocols – Represents a pattern on top of message passing and support two way exchange of messages as encountered on client-server computing.

Remote procedure call – Represents the extended abstraction of a procedure call to distributed environments, so procedure calls on remote machines can be called as if they are procedures in the local address space.

**CLIENT PROCESS**

Client Program <--> Client Stub Procedure <--> Communication Module

🡨 **REQUEST / REPLY** 🡪

**SERVER PROCESS**

Communication Module <--> Dispatcher <--> Server Stub Procedure <-->Service Procedure

**Client stub procedure** acts like a local procedure to the client but instead of executing the call, it marshals the procedure identifier and the arguments into a request message, which it sends via its **communication module** to the server. When the reply message arrives, it unmarshals the results.

The server process contains a **dispatcher**, one **server stub procedure** and one **service procedure for each procedure in the service interface** (remote methods available to client).

The **dispatcher** selects one **service stub procedure** according to the procedure identifier

JSON vs RMI vs Sockets and threading