**Session2**

**Tasks**

**What is the dependency injection?**

Dependency injection is basically providing the objects that an object needs (its dependencies) instead of having it construct them itself. It's a very useful technique for testing, since it allows dependencies to be mocked or stubbed out.

Dependencies can be injected into objects by many means (such as constructor injection or setter injection). One can even use specialized dependency injection frameworks (e.g. Spring) to do that, but they certainly aren't required. You don't need those frameworks to have dependency injection. Instantiating and passing objects (dependencies) explicitly is just as good an injection as injection by framework.

**What are the clean code rules?**

**10 tips for writing clean code in ant programming language**

**1. Use descriptive names**  
**2. Use empty lines to create a readable code**

**3. Do not send more than three parameters into a function**

**4. Remember the functions must do only one thing**

**5. Functions must be small**

**6. Reduce the number of characters in a line**

**7. Avoid using comments**

**8. Create a descriptive message when you create a commit**

**9. Use Unit Test and practice Test Driven Development**

**10. Learn Design Patterns**

**Why recursive code is faster than iterative code?**

This depends on the language being used. You wrote 'language-agnostic', so I'll give some examples.

In Java, C, and Python, recursion is fairly expensive compared to iteration (in general) because it requires the allocation of a new stack frame. In some C compilers, one can use a compiler flag to eliminate this overhead, which transforms certain types of recursion (actually, certain types of tail calls) into jumps instead of function calls.

Threads in Operating System

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.

The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

## **Need of Thread:**

* It takes far less time to create a new thread in an existing process than to create a new process.
* Threads can share the common data, they do not need to use Inter- Process communication.
* Context switching is faster when working with threads.
* It takes less time to terminate a thread than a process.

## **Types of Threads**

In the [operating system](https://www.javatpoint.com/os-tutorial), there are two types of threads.

1. Kernel level thread.
2. User-level thread.

### **User-level thread**

The [operating system](https://www.javatpoint.com/operating-system) does not recognize the user-level thread. User threads can be easily implemented and it is implemented by the user. If a user performs a user-level thread blocking operation, the whole process is blocked. The kernel level thread does not know nothing about the user level thread. The kernel-level thread manages user-level threads as if they are single-threaded processes?examples: [Java](https://www.javatpoint.com/java-tutorial) thread, POSIX threads, etc.

**Advantages of User-level threads**

1. The user threads can be easily implemented than the kernel thread.
2. User-level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3. It is faster and efficient.
4. Context switch time is shorter than the kernel-level threads.
5. It does not require modifications of the operating system.
6. User-level threads representation is very simple. The register, PC, stack, and mini thread control blocks are stored in the address space of the user-level process.
7. It is simple to create, switch, and synchronize threads without the intervention of the process.

**Disadvantages of User-level threads**

1. User-level threads lack coordination between the thread and the kernel.
2. If a thread causes a page fault, the entire process is blocked.

### **Kernel level thread**

The kernel thread recognizes the operating system. There is a thread control block and process control block in the system for each thread and process in the kernel-level thread. The kernel-level thread is implemented by the operating system. The kernel knows about all the threads and manages them. The kernel-level thread offers a system call to create and manage the threads from user-space. The implementation of kernel threads is more difficult than the user thread. Context switch time is longer in the kernel thread. If a kernel thread performs a blocking operation, the Banky thread execution can continue. Example: Window Solaris.

**Advantages of Kernel-level threads**

1. The kernel-level thread is fully aware of all threads.
2. The scheduler may decide to spend more CPU time in the process of threads being large numerical.
3. The kernel-level thread is good for those applications that block the frequency.

**Disadvantages of Kernel-level threads**

1. The kernel thread manages and schedules all threads.
2. The implementation of kernel threads is difficult than the user thread.
3. The kernel-level thread is slower than user-level threads.

## **Components of Threads**

Any thread has the following components.

1. Program counter
2. Register set
3. Stack space

## **Benefits of Threads**

* **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
* **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
* **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
* **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
* **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
* **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.