Multitasking:

- Multitasking is a process of performing multiple tasks simultaneously using single processor.
- We use multitasking to optimize the utilization of CPU.
- Multitasking can be achieved by two ways:
  - Process-based Multitasking(Multiprocessing)
  - Thread-based Multitasking(Multithreading)

Process-based Multitasking (Multiprocessing):

- Each process have its very own location in memory for example each process designates separate memory zone
- Process is heavy weight.
- Cost of communication between the process is high.
- Switching from one process to another (Context-Switching) consumes lot of time.

Thread-based Multitasking (Multithreading):

- Threads share the same address space.
- Thread is lightweight, a smallest unit of processing.
- Cost of communication between the thread is low.
- They don't allocate separate memory area so context-switching between the threads takes less time than processes.

Note:

- At least one process is required for each thread.
- Multithreading is mostly used in games, animation etc.

How to create thread ?

- There are two ways to create a thread:
  - By extending Thread class
  - By implementing Runnable interface

Thread class:

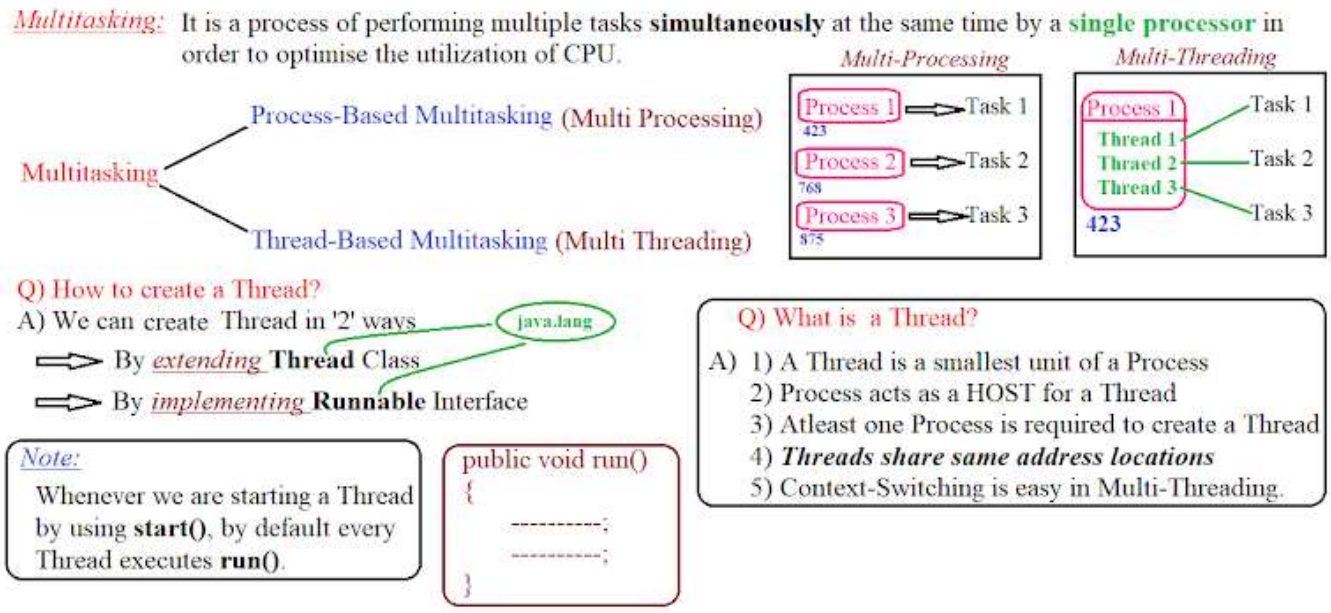
- Thread class is the sub class of 'Object' class and it implements Runnable interface (by default).
- Thread class will be having constructors and methods to perform operations on thread.
- When a class is extending the Thread class, it overrides the run() method from the Thread class to define the code executed by the thread.

Runnable interface:

- Runnable interface will have only one method named run().
- It is mostly recommended to use when creating thread.
- **public void run():** is used to perform action for a thread.

Steps for creating a thread

1) Write a class that extends Thread class or implements Runnable interface this is available in java.lang package.
2) Write public void run () method in that class, this is the method by default executed by any thread.
3) Create an object to that class (Inside main()).
4) Create a Thread Class Object and attach it to your class object.
5) Start running the thread.



We cannot guess the output in threads.

In interview point of multithreading and exception handling have many theoretical questions.

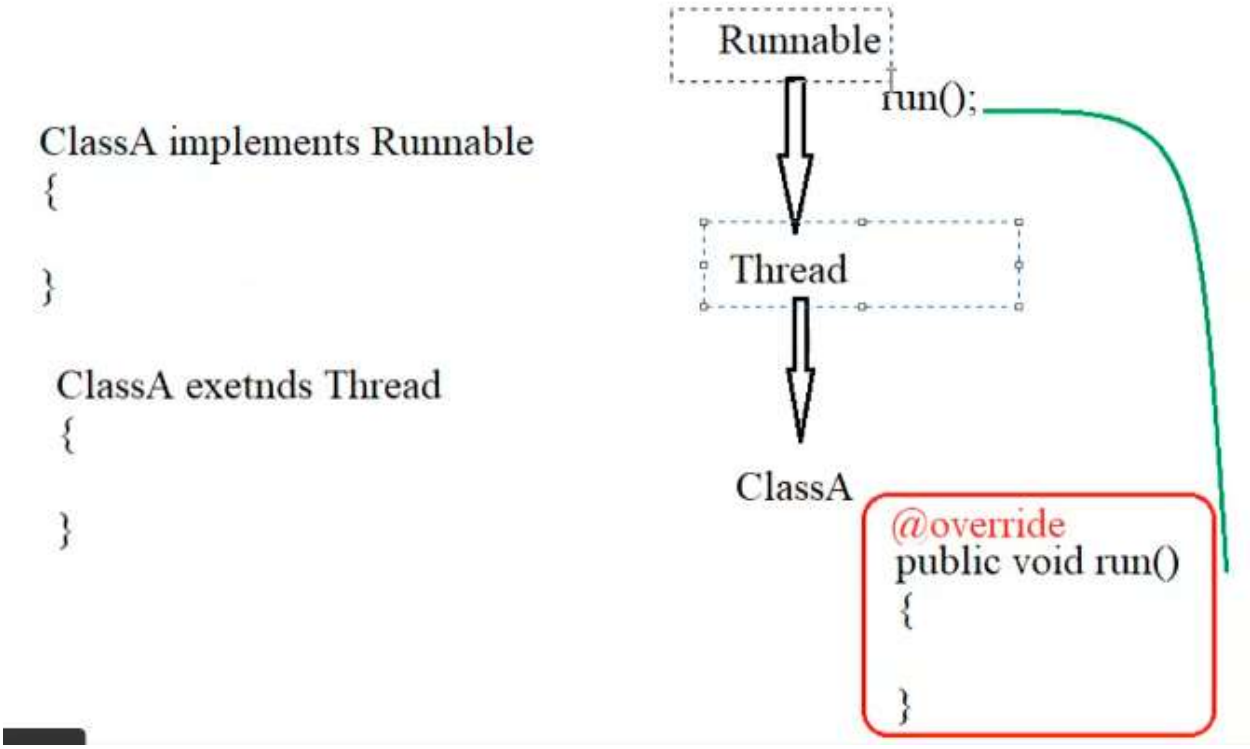
Context-switching is switching the processor from one thread to another thread.

Process is nothing but a program.

Host means without which it cannot survive.

Runnable Interface is a functional interface which has only one abstract method in this we have run().

run() called internally by the start().



Runnable interface is parent class for thread class

Thread-class is the parent class for Class A

This is like multi-level inheritance

Here in above which is better?

Implements Runnable is better than extends Thread

Reason java does not support multiple inheritance through classes

```java
package com.pack1;

public class ClassA extends Thread
{
    @Override
    public void run()
    {
        System.out.println("run() called");
        for(int i=1;i<=5;i++)
        {
            System.out.println("i value : "+i);
        }
        System.out.println("run() execution completed");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        Thread t=new Thread(aobj);
        t.start();

    }
}
```

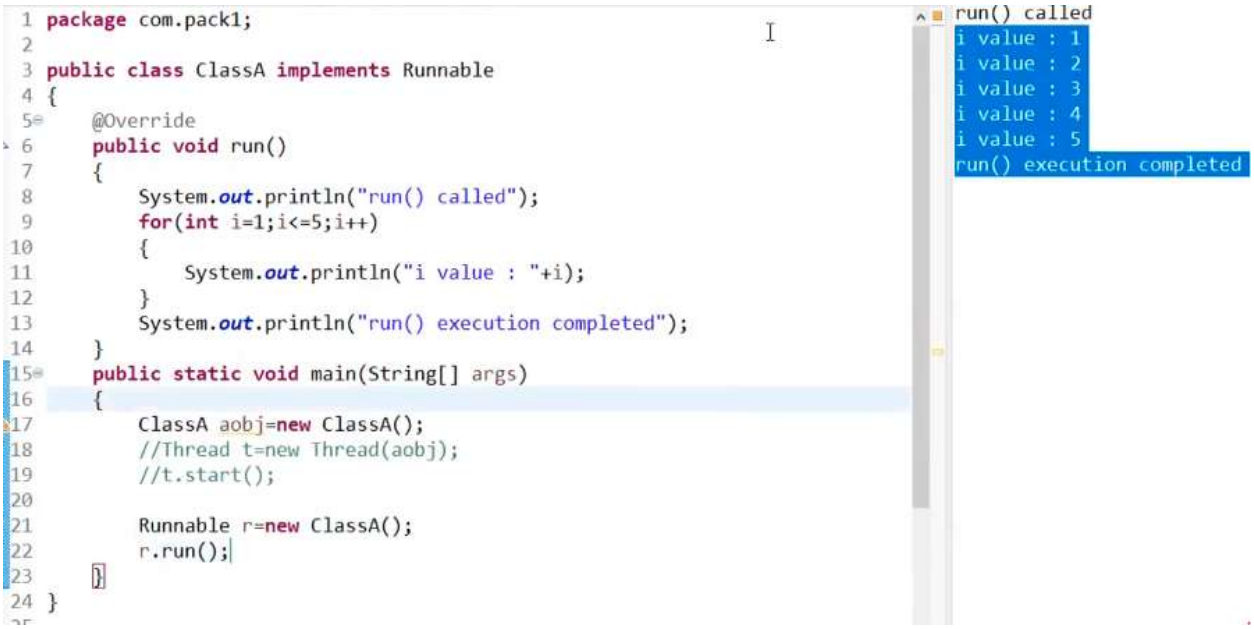
```
run() called
i value : 1
i value : 2
i value : 3
i value : 4
i value : 5
run() execution completed
```

```java
package com.pack1;

public class ClassA implements Runnable
{
    @Override
    public void run()
    {
        System.out.println("run() called");
        for(int i=1;i<=5;i++)
        {
            System.out.println("i value : "+i);
        }
        System.out.println("run() execution completed");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        Thread t=new Thread(aobj);
        t.start();
    }
}
```

```
run() called
i value : 1
i value : 2
i value : 3
i value : 4
i value : 5
run() execution completed
```

Other ways



```java
package com.pack1;

public class ClassA implements Runnable
{
    @Override
    public void run()
    {
        System.out.println("run() called");
        for(int i=1;i<=5;i++)
        {
            System.out.println("i value : "+i);
        }
        System.out.println("run() execution completed");
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        //Thread t=new Thread(aobj);
        //t.start();

        Runnable r=new ClassA();
        r.run();
    }
}
```

```
run() called
i value : 1
i value : 2
i value : 3
i value : 4
i value : 5
run() execution completed
```