Java 8 Interface

```
interface{
abstract methods;
default/ static method body
}
```

```
@FunctionalInterface
interface Test<T>{
T method(T o1, T o2 );//SAM
}
```

```
(x,y)->x+y
Integer:sum;

int sum(int n1,int n2);
```
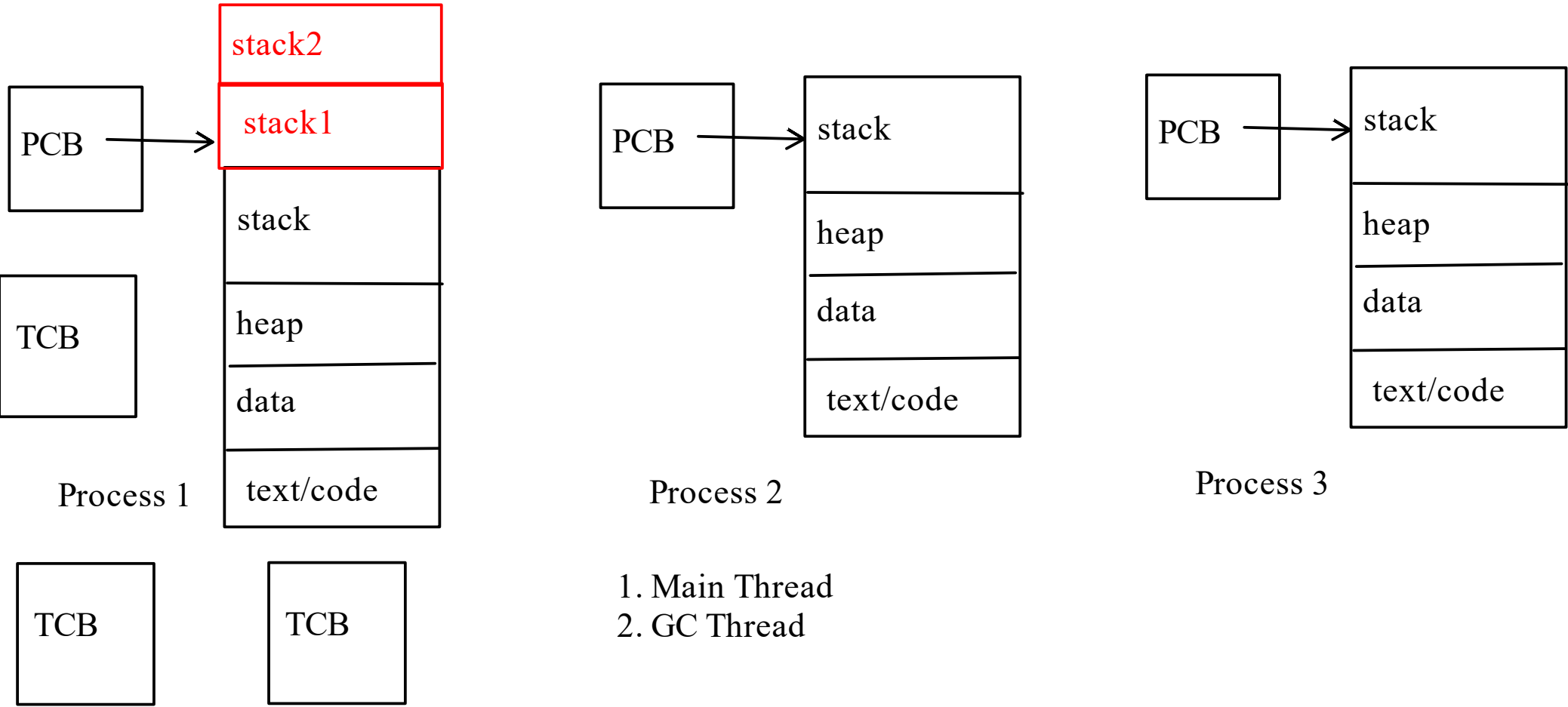
MultiThreading

Operating System
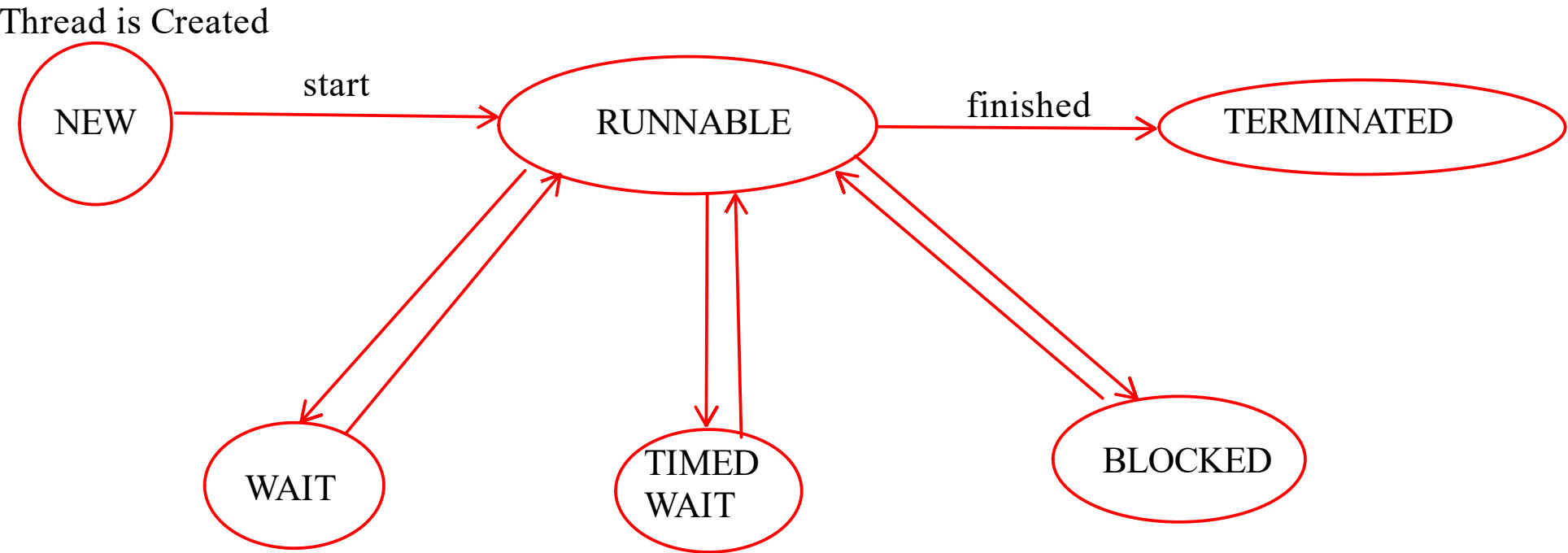Process
    - A Program in execution is called as process

Thread
    - It is a light weight process
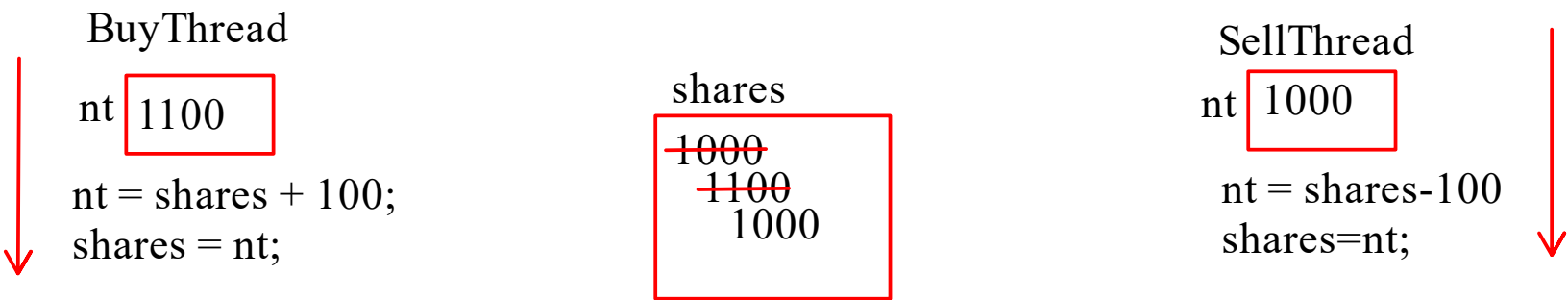   - It is used to perform multiple tasks concurrently within the process

Process 1

Process 2

Process 3

1. Main Thread
2. GC Thread

```
class Employee extends Person implements Runnable{

}

class Employee extends Threadf{

}
Thread th = new Thread(new Employee());
```

All the threads that we create including the main thread are non deamon threads
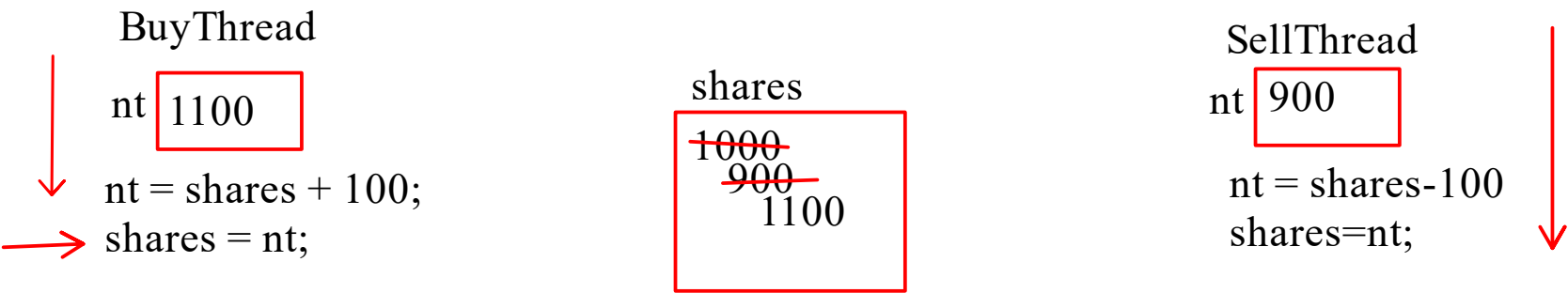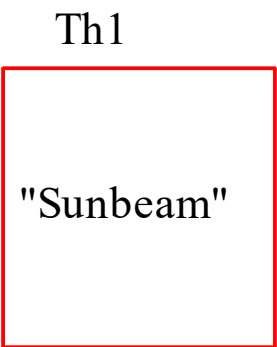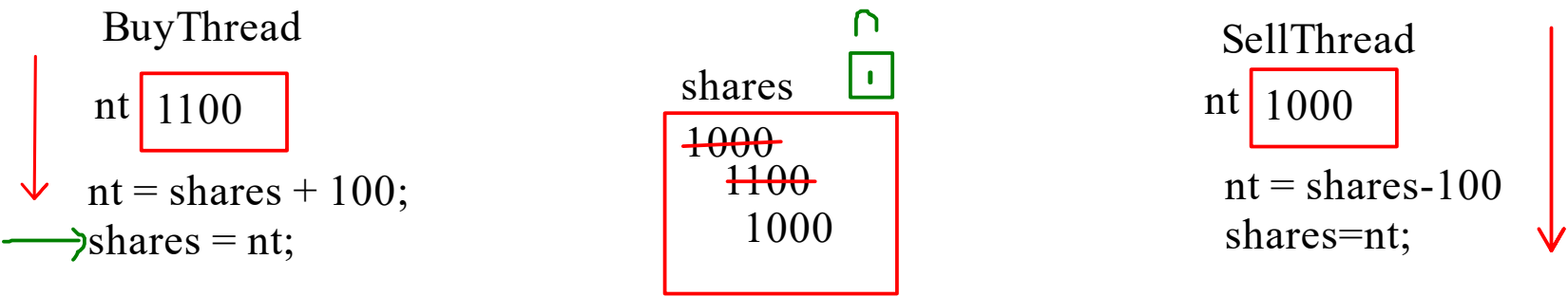These are also called as foreground threads

Thread is Created

NEW → *start* → **RUNNABLE** → *finished* → **TERMINATED**

RUNNABLE ↔ **WAIT**

RUNNABLE ↔ **TIMED WAIT**

RUNNABLE ↔ **BLOCKED**

obj.notify

1. Ideal Condition

BuyThread

nt | 1100 |

nt = shares + 100;
shares = nt;

shares

| ~~1000~~
~~1100~~
1000 |

SellThread

nt | 1000 |

nt = shares-100
shares=nt;

2. Actual Condition

BuyThread

nt | 1100 |

nt = shares + 100;
shares = nt;

shares

| ~~1000~~
~~900~~
1100 |

SellThread

nt | 900 |

nt = shares-100
shares=nt;

3. Synchronization

BuyThread

nt | 1100 |

nt = shares + 100;
shares = nt;

shares

| ~~1000~~
~~1100~~
1000 |

SellThread

nt | 1000 |

nt = shares-100
shares=nt;

Th1

| "Sunbeam" |

obj

Th2

| "Infotech" |
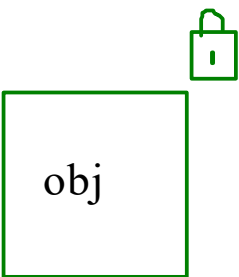
synchrozied(obj){
obj.wait();
System.out.print(name.charAt(i));
delay();
}

th1
th2