# THREADS

*Kamel Rushaidat • Grand Circus Detroit*

# TOPICS

**Topics covered:**

- What are threads
- Threads’ lifecycle

GRAND CIRCUS
·DETROIT·

# THREADS

**Threads**

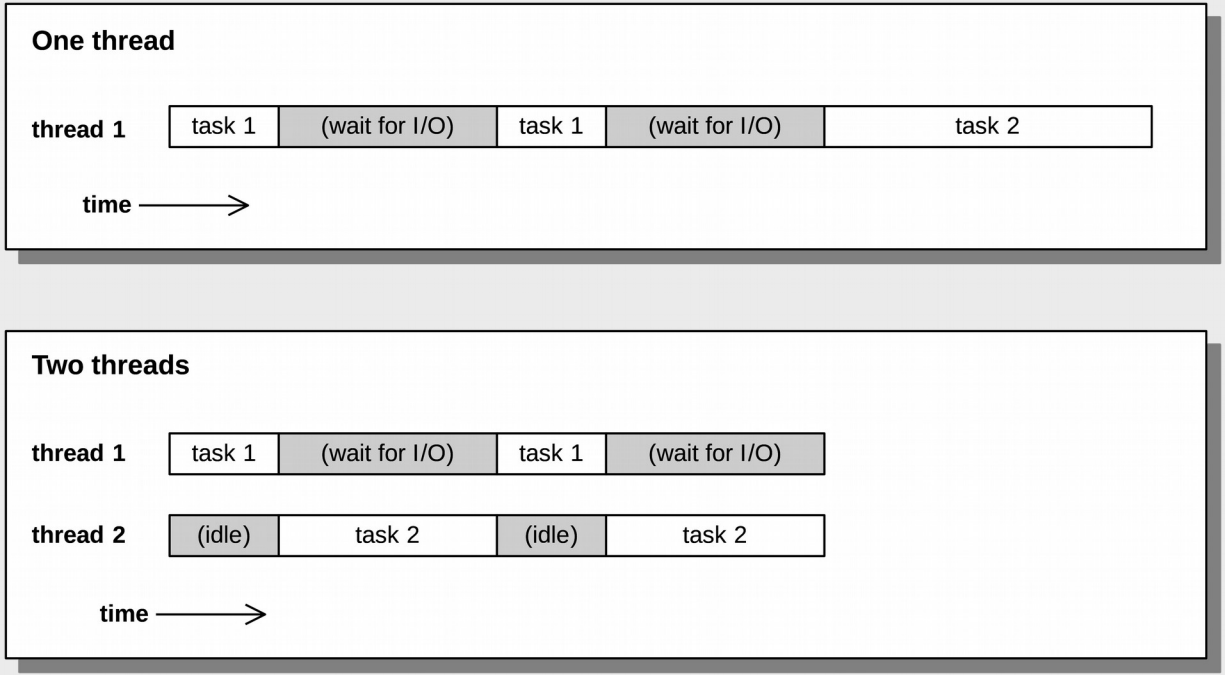
Threads are parts of the program set to run independent of the rest of the program.

We can segregate processor-intensive functions in a Java class to run separately from the rest of the programs using threads.

This is also referred to as multitasking.

# THREADS

## How using threads can improve performance




GRAND CIRCUS
·DETROIT·

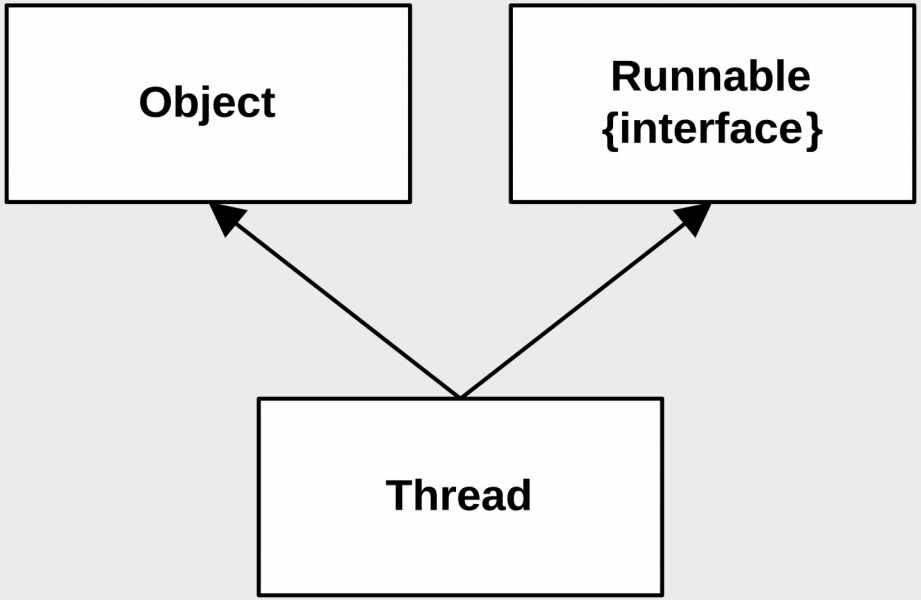
# THREADS

**Typical Uses for Threads**

- To improve the performance of applications with extensive I/O operations
- To improve the responsiveness of GUI applications
- To allow two or more users to run server-based applications simultaneously

# THREADS

**Classes and interfaces used to create threads**



# THREADS

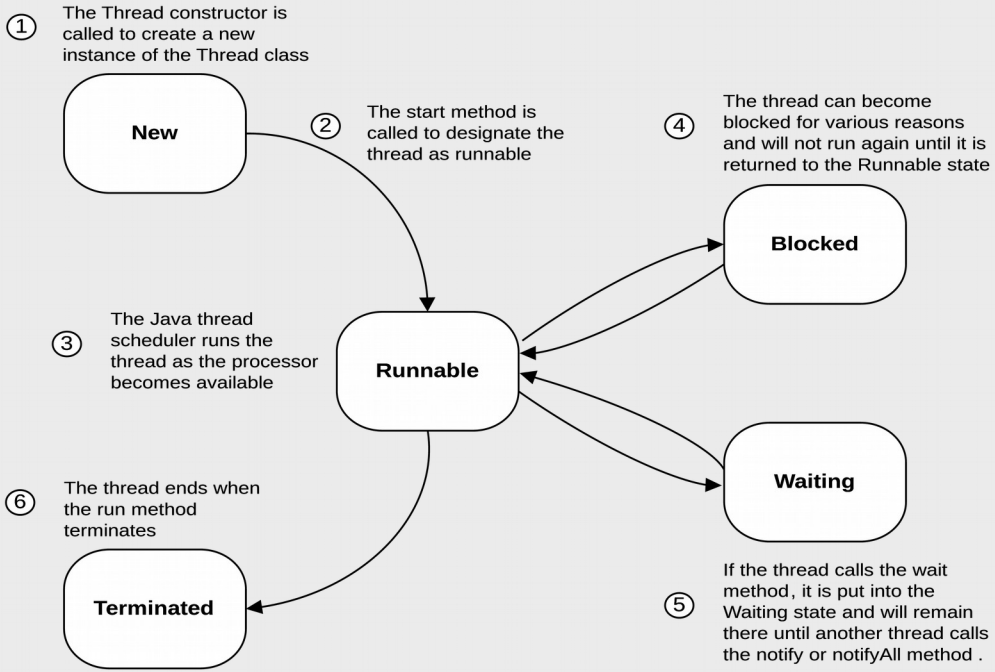
**Two ways to create a thread**

- Inherit the Thread class.

- Implement the Runnable interface, then pass a reference to the Runnable object to the constructor of the Thread class. This is useful if the thread needs to inherit a class other than the Thread class.

# THREADS

## The life cycle of a thread




GRAND CIRCUS
·DETROIT·

# THREADS

**Thread States**

- New
- Runnable
- Blocked
- Waiting
- Terminated

# THREADS

**The Thread class**

java.lang.Thread;

**Common constructors of the Thread class**

- **Thread()**
- **Thread(**Runnable**)**
- **Thread(**String**)**
- **Thread(**Runnable, String**)**

GRAND CIRCUS
·DETROIT·

# THREADS

**Common methods of the Threads class**

- run()
- start()
- getName()
- currentThread()
- setDaemon(boolean)
- sleep(long)
- interrupt()
- isInterrupted()
- yield()

# THREADS

**Creating a thread from the Thread class**

1. Create a class that inherits the Thread class.

2. Override the run method to perform the desired task.

3. Create the thread by instantiating an object from the class.

4. Call the start method of the thread object.

# THREADS

**A Main class that starts a thread**

```
public static void main(String[] args){

    Thread t1 = Thread.currentThread();
    System.out.println(t1.getName() + " started.");

    Thread t2 = new IOThread();  // create the IO thread
    t2.start();  // start the IO thread

    System.out.println(t1.getName() + " starts " +
    t2.getName() + ".");

    System.out.println(t1.getName() + " finished.");
}
```

# THREADS

**Creating a thread using the Runnable interface**

1. Create a class that implements the Runnable interface.

2. Implement the run method to perform the desired task.

3. Create the thread by supplying an instance of the Runnable class to the Thread constructor.

4. Call the start method of the thread object.

# THREADS

**A Main class that starts a thread**

```
public static void main(String[] args){
    Thread t1 = Thread.currentThread();
    System.out.println(t1.getName() + " started.");

    Thread t2 = new Thread(new IOTask());// create the new
                                         //thread
    t2.start; // start the new thread

    System.out.println(t1.getName() + " starts " +
    t2.getName() + ".");
    System.out.println(t1.getName() + " finished.");
}
```

# RECAP

**What you should know at this point:**

- What are threads
- Where are threads used
- How threads can enhance performance
- Classes and interfaces used in thread programming
- Thread lifecycle
- Ways to create and use threads