Многонишково програмиране.

Аделина Алексиева

Клас Thread

- Инстанция на програмата, която се стартира едновременно с други програми се нарича нишка (**thread).**
- Класът **java.lang.Thread** моделира изпълнението на нишката.
- За да създадете нова нишка за изпълнение, трябва да се наследи класа Thread и след това се пише код в метода **run().**
- За да го използвате, трябва да създадете инстанция на класа и да извикате метода **start()** (НЕ метода **run()**).

Конструктори

```
public Thread (Runnable target);
public Thread (String name);
public Thread ();
public Thread (Runnable target,
                        String name);
```

Интерфейс Runnable

- Ако не искате да наследявате Thread, създайте клас, който имплементира интерфейса **Runnable**.
- **Дефинирайте** метода **run()** в този клас.
- Създайте нова **инстанция на Thread,** като предадете вашия **Runnable** клас, като параметър на конструктора и извикайте метода **start()**.

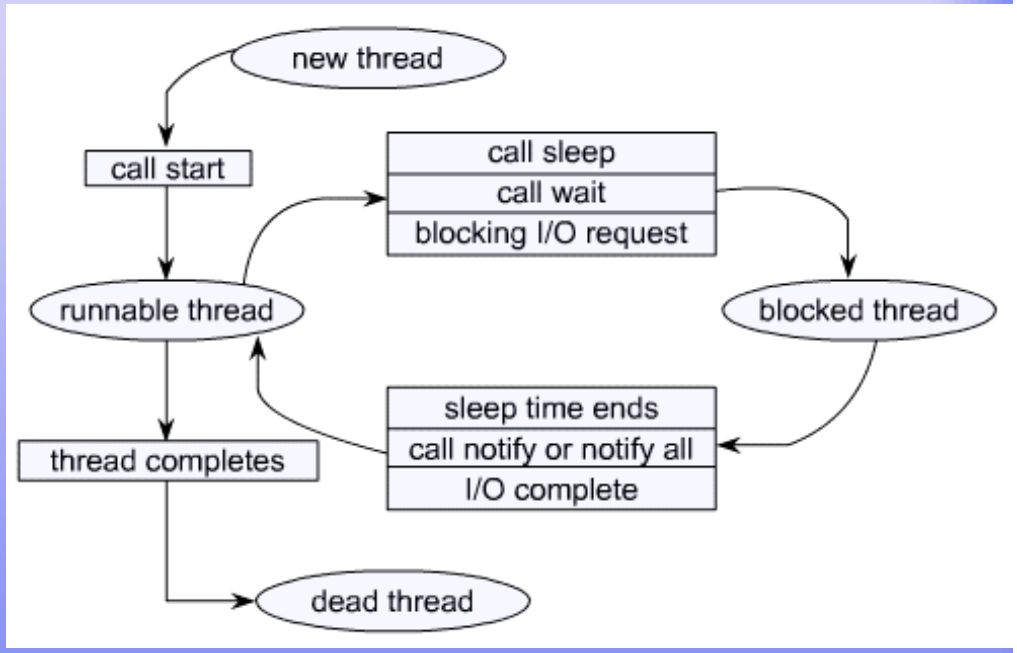
```
public interface Runnable {
     abstract public void run( );
}
```

Създаване и стартиране на нишки

- Създайте клас който наследява класа **Thread**
  - **Пренапишете** метода **run()** с вашата функционалност.
  - Създайте инстанция от вашия нов клас.
  - Извикайте метода **start().**
- Създайте клас, който имплементира **Runnable**
  - Създайте метода **run()** с вашата функционалност.
  - Създайте инстанция от класа **Thread**, като предадете като параметър на конструктора вашия клас.
  - Извикайте метода **start().**

Състояние на нишката

- **Born**
  - Обекта нишка се създава
- **Ready**
  - Методът **start()** се извиква, но това все още не взима време от процесора.
- **Running**
  - Методът **run()** се изпълнява от **JVM.**
- **Blocked**
  - Нишката изчаква да се случи някакво събитие.
- **Dead**
  - Методът run е приключил.



Състояние блокиране

- Позволява на друга нишка да има достъп до процесора.
- Методът **Thread.sleep()** поставя нишката в блокирано състояние за указано време.
- Методът **Thread.join()** указва да изчака изпълнението на другата нишка.

Други методи

- Статичният метод **Thread.currentThread()** получава референция до текущата нишка.
- Методът **isAlive()** установява дали нишката е в състояние **dead** или не.
- Методът **setPriority()** контролира приоритета на нишката.

Синхронизация на нишки

- Понякога различни нишки могат да получат данни до едни и същи данни.
- Две нишки могат да изпълнят един и същи метод по едно и също време.

Синхронизация на нишки

- Група от оператори може да се декларира като **synchronized**
- Само една нишка по едно и също време може да получи достъп до **synchronized** блок.
- Когато две нишки се опитат да получат достъп до един и същ synchronized блок, единия трябва да изчака.

Пример

```
class SpeechSynthesizer {
 synchronized void say(String words){
     // speak
 }
}
```

Демонстрация

- Задача: Да се пуснат две нишки, които да отпечатват съдържанието на два масива и с различно зададено време на изчакване.