Threads
Корректная работа с потоками

Class Thread
Конструкторы

Thread()
Thread(Runnable target)
Thread(Runnable target, String name)
Thread(String name)

Class Thread
Методы

void start() - запускает поток на исполнение. JVM вызовет метод run() потока
long getId() – индитификатор потока
String getName() – имя потока
int getPriority() – приоритет потока
void interrupt() - “просит” поток прерваться
boolean isInterrupted() - проверяет был ли поток прерван
boolean isAlive() – проверяет если поток еще “живой”
booleanisDaemon() – если поток является демон-потоком
void join() – ожидает завершения потока
void join(long millis) – ожидает завершения потока *millis* миллисекунд
void join(long millis, int nanos) – ожидает завершения потока *millis* миллисекунд плюс *nanos* наносекунд
void setDaemon(boolean on) – устанавливает потоку статус демон
void setName(String name) - меняет имя потока
void setPriority(int newPriority) - меняет приоритет потока

Class Thread
Статические методы

static Thread currentThread() – ссылка на текущий поток

static boolean interrupted() - проверить был ли текущий поток прерван, и сбросит флаг если был прерван

static void sleep(long millis) – заставляет поток прекратить свою работу на указанное количество миллисекунд

static void sleep(long millis, int nanos) – заставляет поток прекратить свою работу на указанное количество миллисекунд плюс количество наносекунд

Создание и запуск потоков (extends Thread)

```
class PrimeThread extends Thread {
   PrimeThread() {
   }
   public void run() {
      // compute primes larger than minPrime
   }
}
....

public static void main(String[] args) {
   PrimeThread p = new PrimeThread();
   p.start();
}
```

Создание и запуск потоков (implements Runnable)

```
class PrimeRun implements Runnable {
   PrimeRun() {
   }
   public void run() {
      // compute primes larger than minPrime
   }
}


....

public static void main(String[] args) {
   PrimeRun p = new PrimeRun();
   new Thread(p).start();
}
```

Ожидание завершение потока

```
class PrimeRun implements Runnable {
   PrimeRun() {
   }
   public void run() {
      // compute primes larger than minPrime
   }
}
….

public static void main(String[] args) throws InterruptedException {
   PrimeRun p = new PrimeRun();
   Thread t1 = new Thread(p);
   t1.start();
   t1.join();//CurrentThread will wait finish t1 (blocked method)
   System.out.println("main() is finished");
}
```

Корректное завершение потока (implements Runnable)

```
class PrimeRun implements Runnable {
   public void run() {
      for (int i = 0; i < 10; i++) {
         if(Thread.interrupted()){
            break;
         }
         //hard work
      }
   }
}
....
public static void main(String[] args) throws InterruptedException {
   PrimeRun p = new PrimeRun();
   Thread t1 = new Thread(p);
   t1.join(3000);
   t1.interrupt();
   System.out.println("main() is finished");
}
```

Корректное завершение потока (extends Thread)

```
class PrimeThread extends Thread {
   public void run() {
      for (int i = 0; i < 10; i++) {
         if(isInterrupted()){
            break;
         }
         //hard work
      }
   }
}
....
public static void main(String[] args) throws InterruptedException {
   PrimeThread p = new PrimeThread();
   p.start();
   p.join(3000);
   p.interrupt();
   System.out.println("main() is finished");
}
```