1) Какими значениями инициализируются переменные по умолчанию?

Основные понятия:

**Переменные** (variables) - это контейнеры, которые хранят какие-либо данные.

Виды переменных:

* **Поля** (fields) : переменные, объявленные в классе;
* **Локальные переменные** (local variables) : переменные в методе или в блоке кода;
* **Параметры** (parameters) : переменные в объявлении метода (в сигнатуре).

Переменные по умолчанию не имеют значения по момента их инициализации. То есть, при выполнении следующего кода:

int x;

x++;

программа выдаст ошибку **The local variable x may not have been initialized.** По умолчанию инициализируются переменные класса. Таблица инициализации переменных класса:

| **data type** | **Default value** |
| --- | --- |
| boolean | false |
| char | \u0000 |
| int,short,byte / long | 0 / 0L |
| float /double | 0.0f / 0.0d |
| any reference type | null |

2) Дайте определение понятию “монитор” в контексте многопоточности.

Основные понятия:

Мьютекс — это специальный объект для синхронизации потоков. Он может принимать два состояния: занят и свободен.

Монитор – это дополнительная «надстройка» над мьютексом. Реализация монитора выражена в операторе **synchronized(**оператор предваряет блок кода или метод, который подлежит синхронизации**)**

Задача мьютекса — обеспечить такой механизм, чтобы доступ к объекту в определенное время был только у одного потока.

Монитор представляет своего рода инструмент для управления доступа к объекту. Когда выполнение кода доходит до оператора synchronized монитор объекта res блокируется, и на время его блокировки монопольный доступ к блоку кода имеет только один поток, который и произвел блокировку. После окончания работы блока кода, монитор объекта res освобождается и становится доступным для других потоков. Пример:

public static void main(String[] args) {

        CommonResource commonResource= new CommonResource();

        for (int i = 1; i < 6; i++){

            Thread t = new Thread(new CountThread(commonResource));

            t.setName("Thread "+ i);

            t.start();

        }

    }

class CommonResource{

    int x=0;

}

class CountThread implements Runnable{

    CommonResource res;

    CountThread(CommonResource res){

        this.res=res;

    }

    public void run(){

//ограничиваем доступ к объекту res

        synchronized(res){

            res.x=1;

            for (int i = 1; i < 5; i++){

                System.out.printf("%s %d \n", Thread.currentThread().getName(), res.x);

                res.x++;

                try{

                    Thread.sleep(100);//приостанавливаем поток на 100мсек

                }

                catch(InterruptedException e){}

            }

        }

    }

}

Здесь определен класс CommonResource, который представляет общий ресурс и в котором определено одно целочисленное поле x. Данный код выводит имя потока и используемый ресурс в классе CommonResource.

Когда один поток заходит внутрь блока кода, помеченного словом synchronized, то Java-машина тут же блокирует мьютекс у объекта, который указан в круглых скобках после слова synchronized. Больше ни один поток не сможет зайти в этот блок, пока активный поток его не покинет. Как только наш поток выйдет из блока, помеченного synchronized, то мьютекс тут же автоматически разблокируется и будет свободен для захвата другим потоком. Если же мьютекс был занят, то наш поток будет стоять на месте и ждать когда он освободится.

3) Что является монитором при выполнении участка кода метода?

Монитором является объект, указанный в блоке synchronized участка кода:

 synchronized (synchedList) {

...

                synchedList.wait();

...

        }