1. ***Какие модификации уровня доступа вы знаете, расскажите про каждый из них.***

* *private* (закрытый) — доступ к члену класса не предоставляется никому, кроме методов этого класса. Другие классы того же пакета также не могут обращаться к private-членам.
* *default, package, friendly,* доступ по умолчанию, когда никакой модификатор не присутствует — член класса считается открытым внутри своего собственного пакета, но не доступен для кода, расположенного вне этого пакета.Т.е. если package2.Class2 extends package1.MainClass, то в Class2 методы **без идентификатора** из MainClass видны **не будут**.
* *protected* (защищённый) — доступ в пределах пакета и классов наследников. Доступ в классе из другого пакета будет к методам public и protected главного класса. Т.е. если package2.Class2 extends package1.MainClass, то внутри package2.Class2 методы с идентификатором protected из MainClass будут видны.
* *public* (открытый) — доступ для всех из любого другого кода проекта

Модификаторы в списке расположены по возрастающей видимости в программе.

1. ***Что такое тернарный оператор выбора и как он работает?***

В языке Java есть также специальный тернарный условный оператор, которым можно заменить определённые типы операторов if-then-else — это оператор ?:

Тернарный оператор использует три операнда. Выражение записывается в следующей форме:  
логическоеУсловие ? выражение1 : выражение2  
Если логическоеУсловие равно true, то вычисляется выражение1 и его результат становится результатом выполнения всего оператора. Если же логическоеУсловие равно false, то вычисляется выражение2, и его значение становится результатом работы оператора. Оба операнда выражение1 и выражение2 должны возвращать значение одинакового (или совместимого) типа.

1. ***Дайте определение понятию “взаимная блокировка” в контексте многопоточности.***

Deadlock, он же взаимная блокировка, явление при котором все потоки находятся в режиме ожидания. Чтобы уменьшить шанс появления deadlock’a не рекомендуется использовать методы wait() и notify().

final Object lock1 = new Object();

final Object lock2 = new Object();

//Здесь будет дедлок

    void method01() {

        synchronized(lock1) {

            synchronized(lock2) {

               //doSomething()

            }

        }

    }

void method1() {

        synchronized(lock2) {

           synchronized(lock1) {

               //doSomething()

          }

      }

  }

Чтобы избежать дедлока можно использовать только один блок synchronized, отказаться от wait-notify или использовать такую конструкцию:

//Чтобы избежать дедлока, лучше использовать один lock.

//Если нельзя использовать только один lock, то применяйте такую схему

 void method2() {

     if (identityHashCode(lock1)>=identityHashCode(lock2)) {

         synchronized(lock1) {

             synchronized(lock2) {

               //doSomething()

             }

         }

     }

     else {

         synchronized(lock2) {

             synchronized(lock1) {

                //doSomething()

             }

         }

     }

 }