Уровень 21. Ответы на вопросы к собеседованию по теме уровня.

* **[Блог им. zor07](http://info.javarush.ru/profile/zor07/created/topics/)**

* + [**level21**](http://info.javarush.ru/tag/level21/)
  + , [**конспект**](http://info.javarush.ru/tag/%D0%BA%D0%BE%D0%BD%D1%81%D0%BF%D0%B5%D0%BA%D1%82/)

**1. Перечислите методы класса Object**  
equals()  
hashCode()  
toString()  
getClass()  
notify()  
notifyAll()  
wait()  
wait(long timeOut)  
wait(long timeOut, int nanos)  
  
**2. Зачем нужны методы equals & hashCode?**  
используются для сравнения объектов.  
Цель метода equals – определить идентичны ли объекты внутри, сравнив внутреннее содержание объектов.  
Тк equals работает медленно, сначала сравниваются хэш коды объектов, и, в том случае, если хэш коды равны, идет проверка по equals  
  
**3. Что будет, если переопределить equals, но не переопределить hashCode?**  
изначально hashCode — случайное число.  
Коллекции в Java перед тем как сравнить объекты с помощью equals всегда ищут/сравнивают их с помощью метода hashCode(). И если у одинаковых объектов будут разные hashCode, то объекты будут считаться разными — до сравнения с помощью equals просто не дойдет.  
  
**4. Зачем нужны методы wait, notify, notifyAll?**  
Иногда в программе может оказаться такая ситуация, что нить вошла в блок кода synchronized, заблокировала монитор и не может работать дальше, т.к. каких-то данных еще не хватает: например, файл который она должна обработать еще не загрузился или что-нибудь в таком духе. Для решения этой проблемы и был придуман метод wait(). Вызов этого метода приводит к тому, что нить освобождает монитор и «становится на паузу».  
Для снятия с паузы используются методы notify, notifyAll. Метод notify «размораживает» одну случайную нить, метод notifyAll – все «замороженные» нити данного монитора.  
  
**5. Как правильно клонировать объект?**  
Два типа клонирования.   
Для клонирования объекта по умолчанию нужно:   
а) Добавить интерфейс Cloneable своему классу  
б) Переопределить метод clone и вызвать в нем базовую реализацию:  
  
class Point implements Cloneable

{  
 int x;  
 int y;  
  
 public Object clone()  
 {  
  return super.clone();  
 }  
}

Или можно написать реализацию метода clone самому:

class Point   
{  
 int x;  
 int y;  
  
 public Object clone()  
 {  
  Point point = new Point();  
  point.x = this.x;  
  point.y = this.y;  
  return point;  
 }  
}

**6. Зачем нужен метод finalize() и как он работает?**  
Если ты помнишь, то finalize() – это специальный метод, который вызывается у объекта перед тем, как сборщик мусора его уничтожит.  
Основная цель этого метода – освободить используемые внешние не-Java ресурсы: закрыть файлы, потоки ввода-вывода и т.п.  
  
finalize() работает нестабильно.  
Этот метод не оправдывает возложенных на него надежд. Java-машина может отложить уничтожение объекта, как и вызов метода finalize на сколько угодно. Более того, она вообще не гарантирует, что этот метод будет вызван. В куче ситуаций ради «оптимизации» он не вызывается.  
  
**7. В чем отличие final, finally, finalize?**

* final — модификатор
* Поля не могут быть изменены, методы переопределены
* Классы нельзя наследовать
* Этот модификатор применяется только к классам, методам и переменным (также и к локальным переменным)
* Аргументы методов, обозначенные как final, предназначены только для чтения, при попытке изменения будет ошибка компиляции
* Переменные final не инициализируются по умолчанию, им необходимо явно присвоить значение при объявлении или в конструкторе, иначе – ошибка компиляции
* Если final переменная содержит ссылку на объект, объект может быть изменен, но переменная всегда будет ссылаться на тот же самый объект
* Также это справедливо и для массивов, потому что массивы являются объектами, – массив может быть изменен, а переменная всегда будет ссылаться на тот же самый массив
* Если класс объявлен final и abstract (взаимоисключающие понятия), произойдет ошибка компиляции
* Так как final класс не может наследоваться, его методы никогда не могут быть переопределены

finally — блок в связке try-catch-finally, код в котором выполнится независимо от того вылетело ли исключение в блоке try или нет. Используется для освобождения ресурсов…  
  
finalize- метод в классе Object см 6.  
  
**8. Что такое try-with-resources?**  
Это специальная конструкция try, называемая try-with-resources, в которой Обрати внимание – после try следуют круглые скобки, где объявляются переменные и создаются объекты. Эти объекты можно использовать внутри блока try, обозначенного скобками {}. Когда выполнение команд блока try закончится, независимо от того – нормально оно закончилось или было исключение, для объекта, созданного внутри круглых скобок (), будет вызван метод close();  
  
**9. Чем отличаются методы wait(1000) и sleep(1000)?**  
sleep() приостанавливает поток на указанное. состояние меняется на TIMED\_WAITING, по истечению — RUNNABLE  
  
wait() меняет состояние потока на WAITING   
может быть вызвано только у объекта владеющего блокировкой, в противном случае выкинется исключение IllegalMonitorStateException. при срабатывании метода блокировка отпускается, что позволяет продолжить работу другим потокам ожидающим захватить ту же самую блокировку. в случае wait(int) с аргументом состояние будет TIMED\_WAITING  
  
**10. В чем отличие i++ и ++i ?**  
++i, i сначала увеличивается на 1, затем участвует в выражении.  
i++, i сначала участвует в выражении, затем увеличивается на 1.