ORACLE

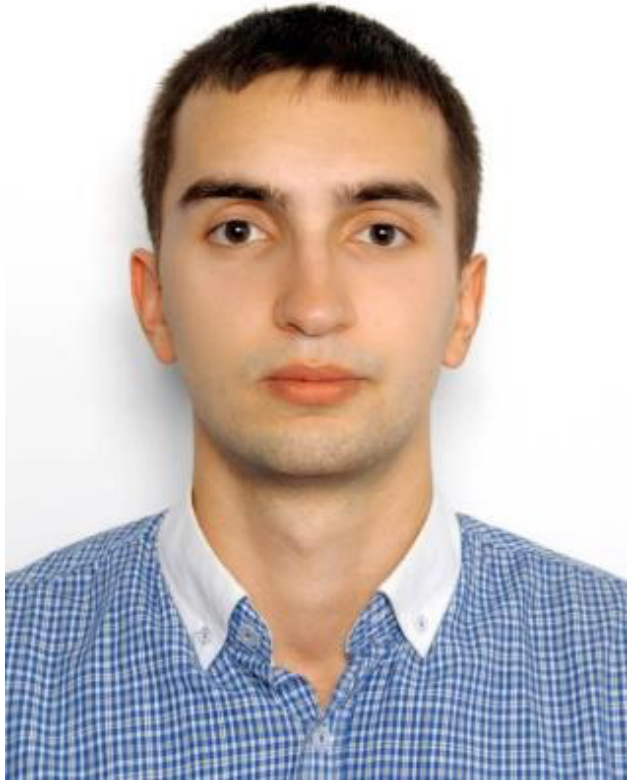
Java Essential

Суперкласс Object

ITVDN
IT VIDEO DEVELOPERS NETWORK

Информационный видеосервис для разработчиков программного обеспечения

Автор курса



Евгений Тихонов

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Доступ можно получить через руководство вашего учебного центра

Проверьте как Вы усвоили данный материал на TestProvider.com

Суперкласс Object

Object

Суперкласс Object

Object – это базовый класс для всех остальных объектов в Java. Каждый класс наследуется от Object. Соответственно все классы наследуют методы класса Object.

Методы класса Object

public final native Class getClass() – возвращает объект класса Class, который описывает класс (имя, методы, поля), от которого был порожден этот объект.

public native int hashCode() – возвращает значение int. Цель хэш-кода – представить любой объект целым числом. Необходимо, чтобы объекты, равные по значению, возвращали одинаковые хэш-коды.

public boolean equals(Object obj) – служит для сравнения объектов по значению, а не по ссылке. Сравнивается состояние объекта, у которого вызывается этот метод, с передаваемым аргументом.

public String toString() – позволяет получить текстовое описание любого объекта. Создавая новый класс, данный метод можно переопределить и возвращать более подробное описание. Для класса Object и его наследников, не переопределивших toString(), метод возвращает следующее выражение:

```
getClass().getName()+"@"+hashCode()
```

Методы класса Object

protected native Object clone() throws CloneNotSupportedException – при выполнении метода clone() сначала проверяется, можно ли клонировать исходный объект. Если разработчик хочет сделать объекты своего класса доступными для клонирования через Object.clone(), то он должен реализовать в своем классе интерфейс Cloneable. В этом интерфейсе нет ни одного элемента, он служит лишь признаком для виртуальной машины, что объекты могут быть клонированы. В ином случае, метод порождает ошибку CloneNotSupportedException.

Если интерфейс Cloneable реализован, то порождается новый объект от того же класса, от которого был создан исходный объект. При этом копирование выполняется на уровне виртуальной машины, никакие конструкторы не вызываются. Затем значения всех полей, объявленных, унаследованных либо объявленных в родительских классах, копируются. Полученный объект возвращается в качестве клона.

Примитивные поля копируются и далее существуют независимо в исходном и клонированном объектах. Изменение одного не сказывается на другом.

Ссылочные поля копируются по ссылке, оба объекта ссылаются на одну и ту же область памяти (исходный объект). Поэтому изменения, происходящие с исходным объектом, сказываются на клонированном.

Методы класса Object

public final native void notify() – возобновляет выполнение потока, который ожидает вызывающего объекта.

public final native void notifyAll() – возобновляет выполнение всех потоков, которые ожидают вызывающего объекта.

public final void wait() throws InterruptedException – переводит выполнение потока в режим ожидания пока не будет вызван метод notify() или notifyAll().

public final native void wait(long timeout) throws InterruptedException – перегруженный метод. переводит выполнение потока в режим ожидания пока не будет вызван метод notify() или notifyAll() или пока не истечет timeout, заданный в миллисекундах.

public final void wait(long timeout, int nanos) throws InterruptedException – перегруженный метод. переводит выполнение потока в режим ожидания пока не будет вызван метод notify() или пока не истечет период времени timeout+nanos, заданный в миллисекундах и наносекундах соответственно.

Методы класса Object

protected void finalize() throws Throwable – данный метод вызывается при уничтожении объекта автоматическим сборщиком мусора (garbage collector). В классе Object он ничего не делает, однако в классе-наследнике позволяет описать все действия, необходимые для корректного удаления объекта, такие как закрытие соединений с БД, сетевых соединений, снятие блокировок на файлы и т.д. В обычном режиме напрямую этот метод вызывать не нужно, он отработает автоматически. Если необходимо, можно обратиться к нему явным образом.

В методе finalize() нужно описывать только дополнительные действия, связанные с логикой работы программы. Все необходимое для удаления объекта JVM сделает сама.

Ключевые слова

Модификатор static

- Применяется к внутренним классам, методам, переменным и логическим блокам.
- Статические переменные инициализируются во время загрузки класса.
- Статические переменные едины для всех объектов класса (одинаковая ссылка).
- Статические методы имеют доступ только к статическим переменным.
- К статическим методам и переменным можно обращаться через имя класса.
- Статические блоки выполняются во время загрузки класса.
- Не static методы не могут быть переопределены как static.
- Локальные переменные не могут быть объявлены как static.
- Абстрактные методы не могут быть static.
- Static поля не сериализуются (только при реализации интерфейса Serializable).
- Только static переменные класса могут быть переданы в конструктор с параметрами, вызывающийся через слово super(//параметр//) или this(//параметр//).

Ключевые слова

Модификатор abstract

1. Применяется только для методов и классов.
2. У абстрактных методов нет тела метода.
3. Является противоположностью final: final класс не может наследоваться, abstract класс обязан наследоваться.

Класс должен быть объявлен как abstract если:

- он содержит хотя бы один абстрактный метод;
- он не предоставляет реализацию наследуемых абстрактных методов;
- он не предоставляет реализацию методов интерфейса, реализацию которого он объявил;
- необходимо запретить создание экземпляров класса.

Модификатор final

- 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.

Ключевые слова

Модификатор strictfp

- Применяется для методов и классов
- Обеспечивает выполнение операций над числами типа **float** и **double** (с плавающей запятой) по стандарту **IEEE 754**

Модификатор transient

- Применяется только для переменных уровня класса (локальные переменные не могут быть объявлены как transient).
- Transient переменные могут не быть final или static.
- Transient переменные не сериализуются.

Модификатор volatile

- Используется только с переменными.
- Может использоваться со static переменными.
- Не используется с final переменными – значение переменной, объявленной как volatile, измененное одним потоком, асинхронно меняется и для других потоков.
- Применяется в многопоточных приложениях.

Ключевые слова

Модификатор synchronized

- Применяется только к методам или частям методов.
- Используется для контроля доступа к важным частями кода в многопоточных программах.

Ключевые слова

Модификатор native

- Используется только для методов.
- Обозначает, что метод написан на другом языке программирования.
- Классы в Java используют много native методов для повышения производительности и доступа к аппаратным средствам.
- Можно предавать/возвращать Java объекты из native методов.
- Сигнатура метода должна заканчиваться “;”, фигурные скобки вызовут ошибку компиляции.

Ключевые слова

Модификатор interface

- Методы всегда **public** и **abstract**, даже если это не объявлено.
- Методы не могут быть **final**, **strictfp**, **native**, **private**, **protected**.
- Переменные только **public static final**, даже если это не объявлено.
- Переменные не могут быть **strictfp**, **native**, **private**, **protected**.
- Может только наследовать (**extends**) другой интерфейс, но не реализовывать интерфейс или класс (**implements**).

Ключевые слова

Сводная таблица

	Класс	Внутренний класс	Переменная	Метод	Конструктор	Логический блок
public	Да	Да (кроме локальных и анонимных классов)	Да	Да	Да	Нет
protected	Нет	Да (кроме локальных и анонимных классов)	Да	Да	Да	Нет
default	Да	Да	Да (и для локальной переменной)	Да	Да	Да
private	Нет	Да (кроме локальных и анонимных классов)	Да	Да	Да	Нет
final	Да	Да (кроме анонимных классов)	Да (и для локальной переменной)	Да	Нет	Нет
abstract	Да	Да (кроме анонимных классов)	Нет	Да	Нет	Нет

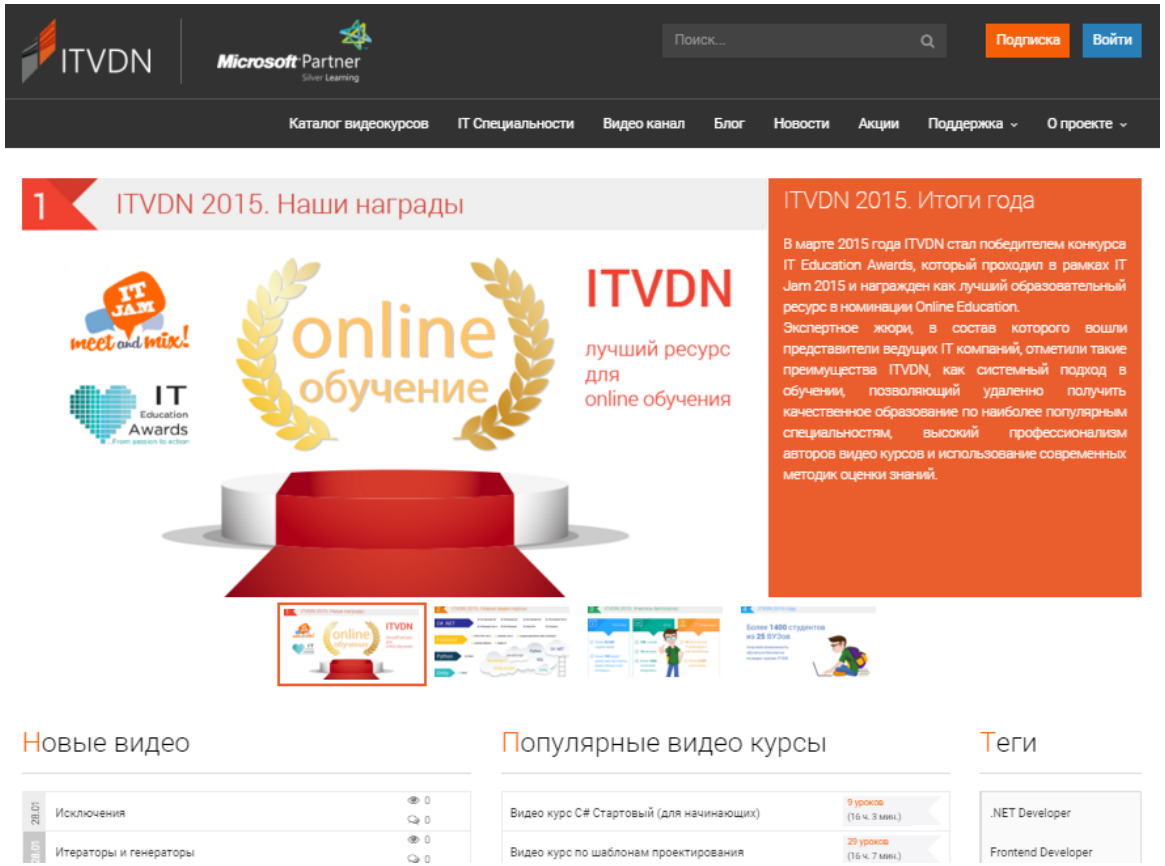
Ключевые слова

Сводная таблица

	Класс	Внутренний класс	Переменная	Метод	Конструктор	Логический блок
static	Нет	Да (кроме локальных и анонимных классов)	Да	Да	Нет	Да
native	Нет	Нет	Нет	Да	Нет	Нет
transient	Нет	Нет	Да	Нет	Нет	Нет
synchronized	Нет	Нет	Нет	Да	Нет	Да (только как часть метода)
volatile	Нет	Нет	Да	Нет	Нет	Нет
strictfp	Да	Да	Нет	Да	Нет	Нет

Смотрите наши уроки в видео формате

ITVDN.com



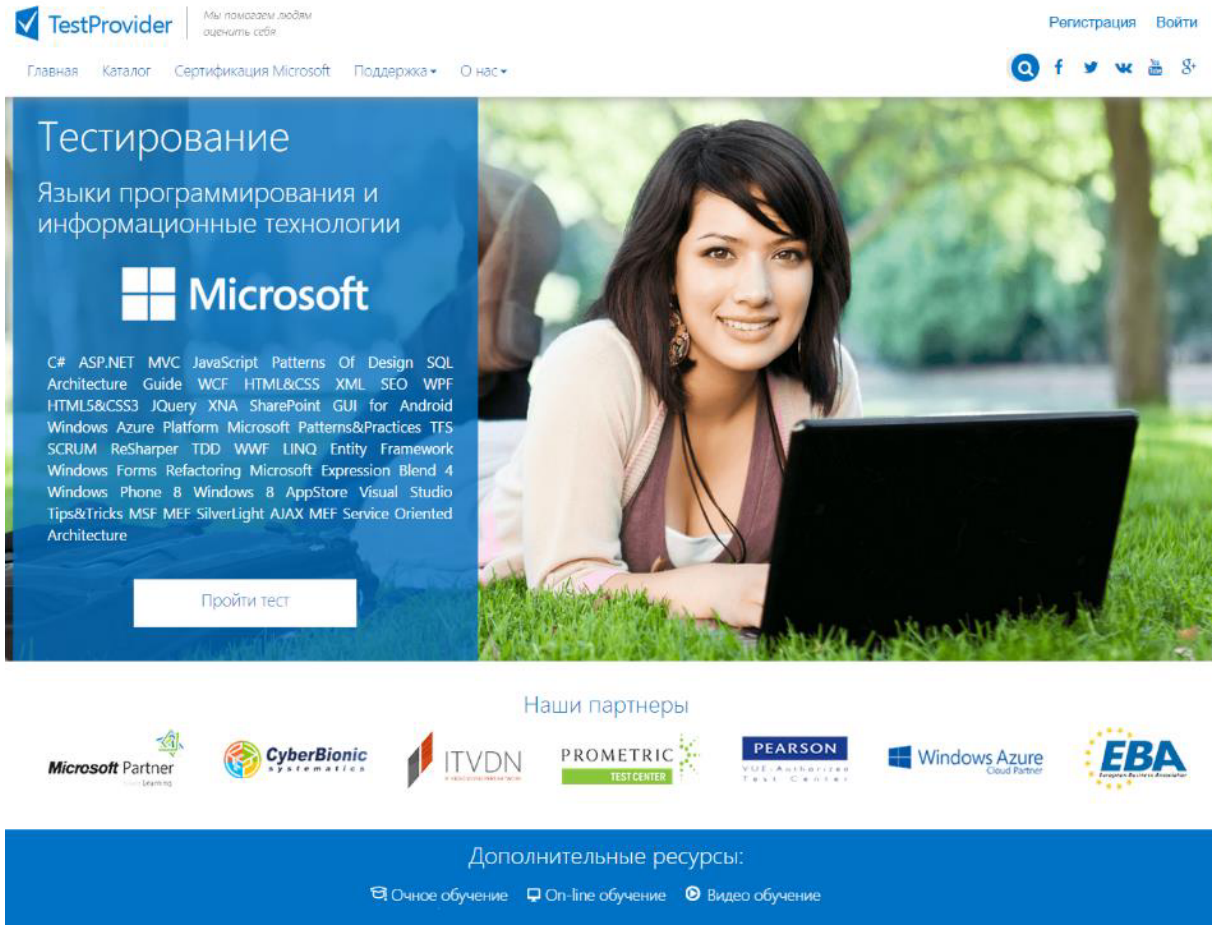
Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A


ITVDN
IT VIDEO DEVELOPERS NETWORK