Программирование на Java

6. Система классов

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Общий базовый класс

- В языке Java ЛЮБОЙ класс является неявным наследником класса Object (иначе говоря, экземпляр любого класса ЯВЛЯЕТСЯ объектом)
- Что такое Object?
  - `public boolean equals(Object o);`
  - `public int hashCode();`
  - `public String toString();`
  - `public Class getClass();`
  - `protected Object clone();`
  - `protected void finalize();`
  - + методы синхронизации потоков (wait / notify)

Методы класса Object

- equals – сравнение двух ЛЮБЫХ ОБЪЕКТОВ на равенство СОДЕРЖИМОГО; по умолчанию – каждый объект равен ТОЛЬКО самому себе
- свойства операции сравнения на равенство:
  - рефлексивность – любой объект ВСЕГДА равен самому себе

Методы класса Object

- equals – сравнение двух ЛЮБЫХ ОБЪЕКТОВ на равенство СОДЕРЖИМОГО; по умолчанию – каждый объект равен ТОЛЬКО самому себе
- свойства операции сравнения на равенство:
  - рефлексивность –
    любой объект ВСЕГДА равен самому себе
  - симметричность –
    если x.equals(y), то y.equals(x) и наоборот
  - транзитивность –
    если x.equals(y) и y.equals(z), то x.equals(z)

Пример

- Сравнение Point / ColoredPoint
  - См. part2.point
- Верно ли, что равны друг другу Point(1, 2) и ColoredPoint(1, 2, 0xff0000) ?

Методы класса Object

- equals – сравнение двух ЛЮБЫХ ОБЪЕКТОВ на равенство СОДЕРЖИМОГО; по умолчанию – каждый объект равен ТОЛЬКО самому себе
- свойства операции сравнения на равенство:
  - рефлексивность – любой объект ВСЕГДА равен самому себе
  - симметричность – если x.equals(y), то y.equals(x) и наоборот
  - транзитивность – если x.equals(y) и y.equals(z), то x.equals(z)
  - никакой объект не равен `null`
  - консистентность

Методы класса Object

- equals – сравнение двух ЛЮБЫХ ОБЪЕКТОВ на равенство СОДЕРЖИМОГО; по умолчанию – каждый объект равен ТОЛЬКО самому себе
- свойства операции сравнения на равенство:
  - рефлексивность – любой объект ВСЕГДА равен самому себе
  - симметричность – если x.equals(y), то y.equals(x) и наоборот
  - транзитивность – если x.equals(y) и y.equals(z), то x.equals(z)
  - никакой объект не равен `null`
  - консистентность
- сравнение на равенство используется в некоторых методах коллекций (в каких?)

Методы класса Object

- hashCode – формирование хэш-кода объекта
  - хэш-коды РАВНЫХ объектов (с точки зрения equals) ДОЛЖНЫ быть равны
  - хэш-коды НЕРАВНЫХ объектов ПО ВОЗМОЖНОСТИ должны различаться; по умолчанию хэш-код равен адресу объекта
- Если в некотором классе переопределен метод equals, НЕОБХОДИМО переопределить метод hashCode

Методы класса Object

- toString – формирование строкового представления объекта; по умолчанию формируется из адреса объекта
- getClass – возвращает объект типа Class, имеющий доступ к спискам полей и методов данного типа (Reflection, рефлексия, интроспекция – отслеживание собственной структуры)

Методы класса Object

- clone() – возвращает копию данного объекта
- finalize() – вызывается сборщиком мусора перед разрушением объекта

Интерфейс Cloneable (JavaDoc)

- public interface **Cloneable**
- A class implements the Cloneable interface to indicate to the Object.clone() method that it is legal for that method to make a field-for-field copy of instances of that class.
- Invoking Object's clone method on an instance that does not implement the Cloneable interface results in the exception CloneNotSupportedException being thrown.
- By convention, classes that implement this interface should override Object.clone (which is protected) with a public method. See Object.clone() for details on overriding this method.

Интерфейс Cloneable

- Не включает в себя ни одной функции (так называемый marker interface)
- Object.clone() проверяет, реализует ли данный объект Cloneable (если нет, бросает исключение)
- Плюс имеется соглашение (неконтролируемое!) о том, что классы, реализующие Cloneable, переопределяют clone() как открытый метод

Интерфейс Comparable<T>

```
public interface Comparable<T> {
    /**
     * @param o the object to be compared
     * @return a negative integer, zero, or
     * a positive integer as this object is
     * less than, equal to, or greater than
     * the specified object
     */
    public int compareTo(T o);
}
```

Что такое <T> и просто T?

- При реализации интерфейса вместо T следует подставить тот тип, с которым мы умеем сравниваться
- В Java, интерфейсы и классы с возможностью подобной настройки называются **generic**
- Тип T – **всегда** ссылочный

Интерфейс Comparator<T>

```
public interface Comparator<T> {
    /**
     * @param o1 1st object to be compared.
     * @param o2 2nd object to be compared.
     * @return a negative integer, zero,
     * or a positive integer as
     * 1st argument is less than, equal to,
     * or greater than 2nd.
     */
    public int compare(T o1, T o2);
}
```

Использование сравнений на неравенство

- См. TreeSet / TreeMap
- Значения должны либо быть попарно Comparable…

Использование сравнений на неравенство

- См. TreeSet / TreeMap
- Значения должны либо быть попарно Comparable…
- … либо в конструктор TreeSet / TreeMap нужно передать Comparator

Использование сравнений на неравенство

- См. TreeSet / TreeMap
- Значения должны либо быть попарно Comparable…
- … либо в конструктор TreeSet / TreeMap нужно передать Comparator
- См. также Collections.sort (тот же принцип)

Generics

- На примере коллекций
  - Collection
  - Collection<?>
  - Collection<? extends T>
  - …

Generics

- Collection = Raw Type = some collection
  - Не рекомендуется в новом коде

Generics

- Collection = Raw Type = some collection
  - Не рекомендуется в новом коде
- Collection<Object>, Collection<String> = Parameterized Type
  - Можно ли присвоить друг другу?

Generics

- Collection = Raw Type = some collection
  - Не рекомендуется в новом коде
- Collection<Object>, Collection<String> = Parameterized Type
  - Можно ли присвоить друг другу? (нет)
- Collection<?> = some collection
  - Безопасный аналог Raw Type

Generics

- Collection = Raw Type = some collection
  - Не рекомендуется в новом коде
- Collection<Object>, Collection<String> = Parameterized Type
  - Можно ли присвоить друг другу? (нет)
- Collection<?> = some collection
  - Безопасный аналог Raw Type
- Collection<? extends Type> = collection of Type or its subtype

Generics

- Collection = Raw Type = some collection
  - Не рекомендуется в новом коде
- Collection<Object>, Collection<String> = Parameterized Type
  - Можно ли присвоить друг другу? (нет)
- Collection<?> = some collection
  - Безопасный аналог Raw Type
- Collection<? extends Type> = collection of Type or its sub-type
- Collection<? super Type> = collection of Type or its **super**-type

Статически вложенные классы

- В некоторых случаях в проекте появляется некоторый класс А, использование которого жестко привязано к классу Б (при этом класс А играет вспомогательную роль)
- Примеры
  - Линейный список и Узел
  - Таблица и Строка таблицы

Спецификаторы вложенных классов

- public – вложенным классом можно пользоваться во всей программе
- private – только во внешнем классе
- protected – во внешнем классе, пакете и наследниках
- static – статически вложенный класс; объект не имеет неявной информации об объекте внешнего класса
  - **если** такая информация не требуется, класс **всегда** следует вкладывать статически

Внутренние и локальные классы

- Внутренние (inner) классы – нестатически вложенные классы
  - объект внутреннего класса несет в себе неявную ссылку на объект внешнего класса (за счет этого можно пользоваться данными и методами внешнего класса)
- Локальные (local) классы – определяются внутри одного из методов внешнего класса

Классы общего назначения из JDK

- Math – содержит ряд математических функций, константы e и pi
- Calendar, Date, TimeZone – работа с датами, временем, поясами
- Random – генератор случайных чисел
- System – методы взаимодействия с системой
- Runtime – методы взаимодействия с JVM

Работа с датами, основные методы

- Calendar c = Calendar.getInstance() – получение объекта-календаря
- c.getTime() – получить время (Date)
- c.getTimeInMillis() – получить время в миллисекундах от 01.01.1970
- c.getTimeZone() – получить временной пояс
- c.setTime(), c.setTimeInMillis(), c.setTimeZone() – установка времени/пояса

Работа со случайными числами, основные методы

- Random r = new Random() – получить генератор
- r.setSeed(Calendar.getInstance().
  getTimeInMillis()) –
  установить стартовое число
- r.nextInt(n) – получить случайное число 0…n–1
- r.nextBoolean() – случайное логическое значение
- r.nextDouble() – случайное вещественное число в интервале [0, 1), равномерное распределение
- r.nextGaussian() – случайное вещественное число, нормальное распределение, матожидание 0.0, среднеквадратичное отклонение 1.0

Взаимодействие с JVM, основные методы

- Runtime r = Runtime.getRuntime() – получение экземпляра Runtime
- r.gc() – принудительно запустить Garbage Collector (сборщик мусора)
- r.totalMemory(), r.freeMemory() – сколько памяти всего использует JVM и сколько ее сейчас свободно
- r.halt(-1), r.exit(-1) – прервать выполнение JVM (второй способ более мягкий)
- r.loadLibrary(libraryName) – загрузить указанную динамическую библиотеку
- r.exec(commandString) – выполнить указанную команду операционной системы

Итоги

- Далее
  - GUI-приложения