СПб АУ НОЦНТ РАН

Java 04

17.03.2017

Static factory methods vs. constructors

```
class MyClass {
    public static MyClass createInstance(...) {
    }

    public MyClass(...) {
    }
}
```

Static factory methods

Свойства возвращаемого объекта

```
BigInteger(int, int, Random) // prime?
BigInteger.probablePrime(int, Random) // prime!
```

Static factory methods

Закэшированные объекты

```
public static Boolean valueOf(boolean b) {
  return (b ? TRUE : FALSE);
}
```

Static factory methods

Подтип объявленного типа

```
Collections.singletone()
Collections.emptyList()
EnumSet.noneOf()
Sets.of()
```

Конструкторы

- Конструкторы супер-класса
- Выделяются в документации, структуре класса и т.п

Классы и интерфейсы. Сходства

```
abstract class A {
    abstract public void foo();
}

interface A {
    void foo();
}
```

- Если есть значение “A a”, можно написать “a.foo()”

Классы и интерфейсы. Сходства

```
abstract class A {
    abstract public void foo();
}

interface A {
    void foo();
}
```

- Если есть значение “A a”, можно написать “a.foo()”
- Нельзя написать “new A()”

Классы и интерфейсы. Сходства

```
abstract class A {
    abstract public void foo();
}

interface A {
    void foo();
}
```

- Если есть значение “A a”, можно написать “a.foo()”
- Нельзя написать “new A()”
- Отделили реализацию от абстракции и радуемся

Технические отличия

	Абстрактные классы	Интерфейсы
Модификаторы методов	public / protected	Неявно abstract public
Не abstract методы	Да	Нет
Состояние (поля)	Да	Нет
Множественное наследование	Нет	Да
Производительность	??	??

Технические отличия

	Абстрактные классы	Интерфейсы
Модификаторы методов	public / protected	Неявно abstract public
Не abstract методы	Да	Нет
Состояние (поля)	Да	Нет
Множественное наследование	Нет	Да
Производительность	??	??

Вывод

Технически классы более гибкие во всем кроме множественного наследования.

Сравнение с C++

- В C++ нет интерфейсов, как отдельной синтаксической конструкции

Сравнение с C++

- В C++ нет интерфейсов, как отдельной синтаксической конструкции
- Классы C++ технически умеют все тоже самое, что и классы в Java

Сравнение с C++

- В C++ нет интерфейсов, как отдельной синтаксической конструкции
- Классы C++ технически умеют все тоже самое, что и классы в Java
- *+ множественное наследование*

Сравнение с C++

- В C++ нет интерфейсов, как отдельной синтаксической конструкции
- Классы C++ технически умеют все тоже самое, что и классы в Java
- *+ множественное наследование*
- => Классы C++ можно использовать, как интерфейсы

Сравнение с C++

- В C++ нет интерфейсов, как отдельной синтаксической конструкции
- Классы C++ технически умеют все тоже самое, что и классы в Java
- *+ множественное наследование*
- => Классы C++ можно использовать, как интерфейсы
- **Минусы?** Множественное наследование с состоянием — очень сложная концепция (вспомним “class A : public virtual B”)
- Можно «примешивать» реализацию (mixins, java 8 default)

Концептуальные отличия

- Интерфейсы определяют поведение/контракт/признак, которым должен удовлетворять объект “x”

Концептуальные отличия

- Интерфейсы определяют поведение/контракт/признак, которым должен удовлетворять объект “x”
- Классы — в частности то же самое

Концептуальные отличия

- Интерфейсы определяют поведение/контракт/признак, которым должен удовлетворять объект “x”
- Классы — в частности то же самое
- Можно выделять общие части в базовые классы
- Классы — непересекающиеся множества

Примеры: LinkedList, List, BaseWindow, Drawable, HashSet, AbstractList, Named, Queue, CharSequence

Возможный ответ

- **Классы**: LinkedList (implements Queue, List), BaseWindow, HashSet, AbstractList
- **Интерфейсы**: Queue, Drawable, Named, CharSequence, List

Naming conventions:

- Классы: *Impl, Base*, Abstract*
- Интерфейсы: *ble, *ed

Логгирование

Варианты использования

- Серверные приложения
- Задачи, выполняющиеся в фоне
- Многопоточность
- Сложная логика, алгоритмы

Логгирование

Подходы

- System.err.println() :(
- java.util.logging
- Log4J
- Logback
- SLF4J

Log4J

```
class A {
  private static final Logger LOG =
      Logger.getLogger(A.class);
  void foo() {
    LOG.warn("my log message");
  }
}
```

Log4J. main/resources/log4.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration>

  <appender name="console" class="org.apache.log4j.ConsoleAp
    <param name="Target" value="System.out"/>
    <layout class="org.apache.log4j.PatternLayout\">
      <param name="ConversionPattern" value="%p %c: %m%n"/>
    </layout>
  </appender>
  <root>
    <priority value ="debug" />
    <appender-ref ref="console" />
  </root>
</log4j:configuration>
```

Уровни логгирования

- FATAL - произошла фатальная ошибка - у этого сообщения наивысший приоритет
- ERROR - в программе произошла ошибка
- WARN - предупреждение в программе что-то не так
- INFO - информация. К сведению трудящися.
- DEBUG - детальная информация для отладки
- TRACE - наиболее полная информация. трассировка выполнения программы. Наиболее низкий приоритет.

Appenders

- FileAppender
- DailyRollingFileAppender
- RollingFileAppender
- SMTPAppender

Log4J

```
LOG.debug("Something " + myObj.toString());
```

Log4J

```
if(LOG.isDebugEnabled() {
  LOG.debug("Something " + myObj.toString());
}
```

Log4J 2

```
private static final Logger LOG =
    LogManager.getLogger(MyApp.class);
LOG.debug("Something {} \%d" myObj, myInt);
```

Log4J 2

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN\">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT\">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5leve
    </Console>
  </Appenders>
  <Loggers>
    <Logger name="com.foo.Bar" level="trace\">
      <AppenderRef ref="Console"/>
    </Logger>
    <Root level="error\">
      <AppenderRef ref="Console\"/>
    </Root>
  </Loggers>
</Configuration>
```

Log4J 2

https://logging.apache.org/log4j/2.0/manual/configuration.html
https://logging.apache.org/log4j/2.0/faq.html

Задание

- На основе ветки master из https://github.com/java-course-au/assignments-2017-1 настроить логгирование (с примером), чтобы сообщения уровня error писались в консоль, а info и выше в файл.
- Прислать результаты в виде PR/на почту Семену
- Дедлайн: 24.03.2017 23:59