More sophisticated behaviour
Lecture 8

Waterford Institute of Technology

May 17, 2014

John Fitzgerald

The Java Library

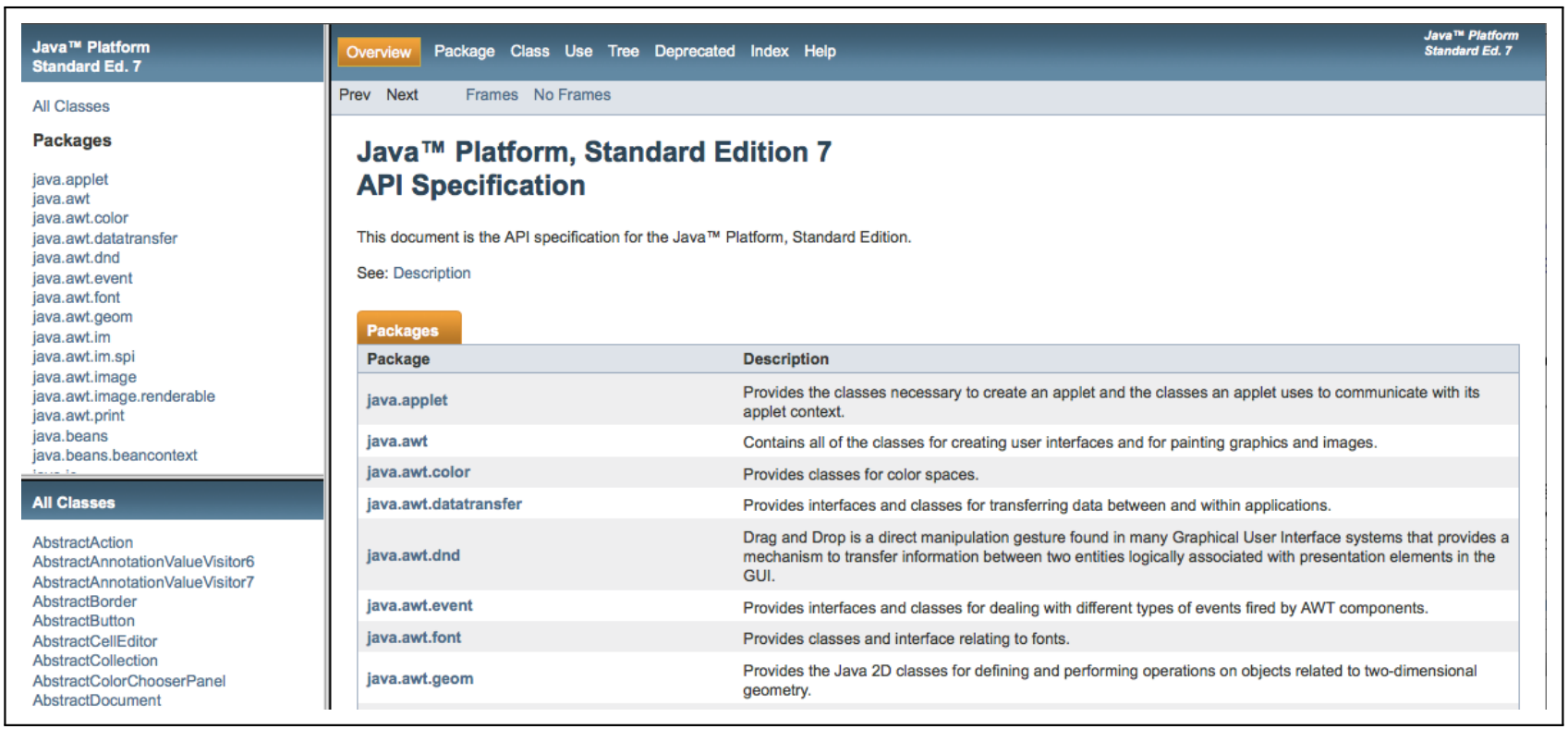
Java library contains thousands of classes

Become familiar with small, frequently used subset

- Classes already encountered
  - String
  - ArrayList
  - Iterator
- Classes explored in this session
  - Random
  - HashMap
  - HashSet
  - Arrays

The Java Library

Overview Java Platform Standard Edition 7



Generics documentation

Parameterized or *Generic classes*

- `Class ArrayList<E>`
  - Array containing objects class type E
  - E specified when ArrayList variable declared
- `Class HashMap<K, V>`
  - K key-type & V mapped value type specified when HashMap variable declared

```
public class GenericsDemo
{
    private ArrayList<String> notes = new ArrayList<String>();
    private HashMap<String,String> contacts = new HashMap<String,String>()
        ;
    public void generics(){
        notes.add("Mustn't forget to call supervisor");
        contacts.put("Abamo Patrick", "(412) 9888 5467");
    }
}
```

Class components

Class may be described as comprising

- Interface
  - Facilitates third party usage
- Implementation
  - Hidden from user
  - Generally user has no interest

```
public class Square
{
    int size;
    public Square(int size) {//interface
        //implementation
        this.size = size;
    }
    public int getArea() {//interface
        //implementation
        return size*size;
    }
}
```

Interface of Class

Comprises following class information

- Name
- General description of purpose
- List constructors
- List methods
- Parameters of constructors and methods
- Return types methods
- Description purpose each constructor and method

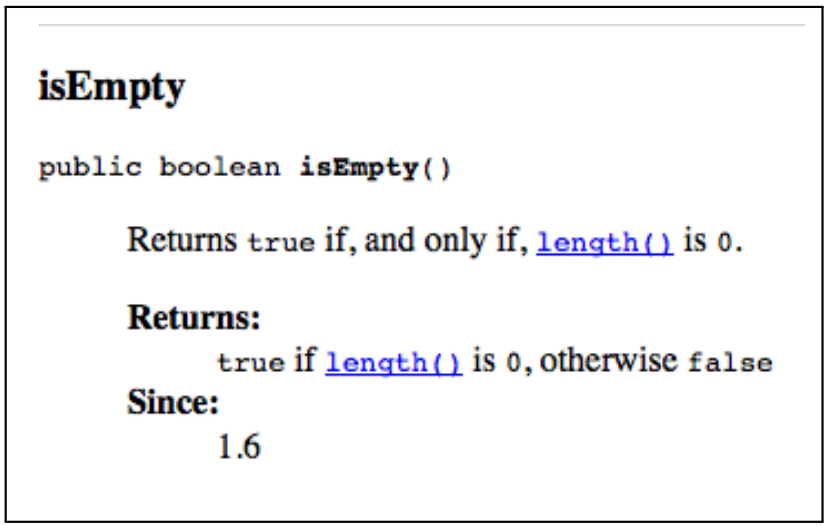
Interface of Method

Interface terminology applicable to methods

Comprises

- Signature
- Descriptive comments

For example `String.isEmpty()`



isEmpty

`public boolean isEmpty()`

Returns `true` if, and only if, `length()` is `0`.

Returns:

`true` if `length()` is `0`, otherwise `false`

Since:

1.6

Implementation of Class

Underlying class source code excluding interface

Implementation

- Important concept
- Generally developer not interested
  - Exceptions: e.g. more efficient version
- Often inaccessible e.g. proprietary code

Wrapper class

Autoboxing

- Applies to wrapper classes such as Integer, Byte and so on
- Eight wrapper classes Java 7
- Example: automatic conversion from primitive *int* to *Integer*
- Known as **autoboxing**
- Useful as keys in collections such as HashMap where primitives not allowed

```
ArrayList<int> values: //not allowed

ArrayList<Integer> values = new ArrayList<>();
for (int i = 1; i < 10; i += 1) {
    values.add(i);
}
```

Wrapper class

Unboxing

- Example: automatic conversion from *Integer* object to *int*
- Known as **Unboxing**

```
ArrayList<Integer> values = new ArrayList<>();
for (int i = 1; i < 10; i += 1) {
    values.add(i);
}
int[] ar = new int[values.size()];
for(int i = 0; i < ar.length; i += 1)
{
  ar[i] = values.get(i);
  System.out.println(ar[i]);
}
```

Java Random class

Objects of Random class can

- Generate pseudorandom number stream
- In range
  - Integer.MIN_VALUE to Integer.MAX_VALUE
  - -2147483648 to 2147483647

```
import java.util.Random;
Random randomGenerator = new Random();
//Generated randNmr is in range −2147483648 to 2147483647
int randNmr = randomGenerator.nextInt();
System.out.println(randNmr);
//Generated randNmr2 is in range 0 to n−1 inclusive
int randNmr2 = randomGenerator.nextInt(n);
```

HashMap

HashMap object that maps keys to values

- Iteration ordering not guaranteed
- Cannot contain duplicate keys
- Each key maps to at most one value
- Has methods such as
  - `put(K key, V value)`
  - `get(Object key)`
  - `containsKey(Object value)`
  - `remove(Object key)`

```
import java.util.HashMap;

contacts.put("DCU", "(353) 1 8658934");
String phoneNumber = contacts.get("DCU");
boolean hasKey = contacts.containsKey("DCU");
contacts.remove("DCU");
```

HashSet

HashSet object has collection distinct elements

- Iteration ordering not guaranteed
- Cannot contain duplicate elements
- Has methods such as
  - `add(E e)`
  - `contains(Object o)`
  - `remove(Object o)`

```
import java.util.HashSet;

names.add("DCU");
names.add("DCU");//ignored
names.contains("DCU");
names.remove("DCU");
```

Arrays

Two dimensional

As with one-dimensional arrays:

- Stores fixed number of elements
- All values same type
- Size fixed at creation

Example creation and initialization 2-d array:

```
int nmrRows = 3;
int nmrCols = 4;
int[][] ar2d = new int[nmrRows][nmrCols];
for(int row = 0; row < nmrRows; row += 1)
{
    for(int col = 0; col < nmrCols; col += 1)
    {
        ar2d[row][col] = row + col;
    }
}
```

0	1	2	3
1	2	3	4
2	3	4	5

Arrays

Two dimensional

Rows may be different lengths

- Each row a one-dimensional array

Example 2-d array variable row lengths:

```
int nmrRows = 3;
int nmrCols = 4;
int[][] ar2d = new int[nmrRows][];
for(int row = 0; row < nmrRows; row += 1)
{
    ar2d[row] = new int[nmrCols + row];
    for(int col = 0; col < ar2d[row].length; col += 1)
    {
        ar2d[row][col] = row + col;
    }
}
```

```
0 1 2 3
1 2 3 4 5
2 3 4 5 6 7
```

Importing Java packages

Use import `qualified-class-name`

Example

- import `java.util.ArrayList;`
- import `java.util.Random;`

Also could use package name but disadvantage possibly thousands classes imported

- import package-`name*;`
- import `java.util*;`

Best be specific

- import `java.util.Date;`
- import `java.util.Random;`

Anonymous objects

Use of anonymous objects common idiom

```
//Verbose
public class College
{
    private String student;

    public College()
    {
        Student student = new Student();
        setState(student);
    }

    public void setState(Student student)
    {
        this.student = student;
    }
}
```

```
//Use anonymous object
public class College
{
    private String student;

    public College()
    {
        setState(new Student());
    }

    public void setState(Student
         student)
    {
        this.student = student;
    }
}
```

Chaining

Chaining method calls

- `setName(name).setAge(age)`
- Executed left to right
- `setName` returns *this*
- Second method call effectively `this.setAge(age)`

```
public class Person {
    private String name;
    private int age;

    public Person(String name, int age)
    {
        setName(name).setAge(age);
    }

    public Person setName(String name)
    {
        this.name = name;
        return this;
    }

    public Person setAge(int age)
    {
        this.age = age;
        return this;
    }
}
```

Control flow

The switch statement

switch statement

- Can have number execution paths
- Execution route depends on value of variable or expression

```
int day = 6;
String sDay;
switch(day)
{
    case 6:
    case 7: sDay = "a weekend day";
            break;
    default: sDay = "a work day";
            break;
}
//Outputs: Today is a weekend day
System.out.println("Today is " + sDay);
```

Control flow

The *break* statement

break statement

- Terminates *for, while, do-while* loop
- Can be labelled or unlabelled

//Example unlabelled *break*

```
int[] arInt = {10, 20, 30, 40, 50, 60};
int searchNmr = 30;
for(int i = 0; i < arInt.length; i += 1) {
    if(arInt[i] == searchNmr) {
        System.out.println("Found it");
        break;
    }
}
```

Control flow

The *continue* statement

continue statement

- Skips current iteration *for, while, do-while* loop
- Unlabelled form
  - skips to end innermost loop's body
  - evaluates boolean expression controlling loop

//Outputs 6. Comment out *continue*: outputs 27

```
String searchMe = "picked peck pickled peppers";
int max = searchMe.length();
int numberPs = 0;

for (int i = 0; i < max; i++) {
    // only count when p found
    if (searchMe.charAt(i) != 'p') {
        continue;
    }
    numberPs++;
}
System.out.println("Found " + numberPs + " p's in the string.");
```

Referenced Material

1. Overview Java Platform Standard Edition 7 `http://docs.oracle.com/javase/7/docs/api/`

[Accessed 2014-02-18]

2. String(Java Platform SE7) `http://docs.oracle.com/javase/7/docs/api/java/lang/String.html`

[Accessed 2014-02-19]

3. ArrayList(Java Platform SE 7) `http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html`

[Accessed 2014-02-19]

4. Random(Java Platform SE 7) http://docs.oracle.com/javase/7/docs/api/java/util/Random.html

[Accessed 2014-02-19]

5. HashMap(Java Platform SE 7) http://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html

[Accessed 2014-02-20]

6. HashSet(Java Platform SE 7) http://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html

[Accessed 2014-02-20]

7. AutoBoxing[Java Platform SE 7) (http://docs.oracle.com/javase/tutorial/java/data/autoboxing.html)

[Accessed 2014-02-24]