Packages, Characters, Strings
Arguments to method main

CS2110, Week 2 Recitation

Package
Package: A collection of Java classes and other packages. See JavaSummary.pptx, slide 20
Available in the course website in the following location:

1. Java classes that are contained in a specific directory on your hard drive (it may also contain sub-packages) or
2. Packages of Java classes that come with Java, e.g. packages java.lang, javax.swing.

Consider first the packages that come with Java. We show you:
(1) How to refer to them
(2) How to find out how to use them, using the API (Application Programmer Interface) specifications.

API packages that come with Java
Visit course webpage, click Links, then Java 1.7 API Specs.
Link: http://www.cs.cornell.edu/courses/CS2110/2014sp/links.html
Scroll down in left col (Packages pane), click on java.lang

Specs for Class Character
Main pane now contains description of class Character:
1. The header of its declaration.
2. A description, including info about Unicode
3. Nested class summary (skip it)
4. Field summary (skip it)
5. Constructor summary (read)
6. Method summary (read)
7. Field detail (skip it)
8. Method detail (read)

Find method compareTo
See a 1-sentence description
Click on method name
Takes you to a complete description in Method detail section

More on class Character later

Package java.lang vs. other packages
You can use any class in package java.lang. Just use the class name, e.g.
Character
To use classes in other API packages, you have to give the whole name, e.g.
javax.swing.JFrame

So you have to write:
javax.swing.JFrame jf= new javax.swing.JFrame();
**Use the import statement!**

To be able to use just `JFrame`, put an import statement before the class definition:

```
import javax.swing.JFrame;
```

**Hard drive**

Eclipse

- Hashing
- recitation02
- src
  - Rec02.java
  - Rec02Tester.java
  - pack1
  - C.java

Eclipse Package Explorer

- **Rec02**
  - src
    - Rec02.java
    - Rec02Tester.java
    - pack1
    - C.java

Eclipse does not make a directory for the default package; its classes go right in directory `src`.

**Primitive type char**

```
Use single quotes

char fred = 'a';
char wilma = 'b';
System.out.println(fred);
```

**Special chars** worth knowing about

- `\t` - tab character
- `\n` - newline character
- `'` - single quote character
- `"` - double quote character
- `\` - backslash character
- `\b` - backspace character - NEVER USE THIS
- `\f` - formfeed character - NEVER USE THIS
- `\r` - carriage return - NEVER USE THIS

Backslash, called the escape character.
Casting char values

Cast a char to an int using unary prefix operator (int).
Gives unicode representation of char, as an int

(int) 'a'  gives 97
(char) 97  gives 'a'

Relational operators are used in a relation or in arithmetic, a char automatically cast to type int.

Relations  <  <=  >=  ==  !=

'c' < 'b'  same as  97 < 98, i.e. false
'c' + 1  gives 98

Class Character

- Each instance of class Character wraps an int value — has a field that contains an int value. Allows a char value to be treated as an object.
- Find methods in each object by looking at API specs on web:
  docs.oracle.com/javase/7/docs/api/java/lang/Character.html

  c.charValue()  c's wrapped char, as a char
  c.equals(c1)  True iff c1 is a Character and wraps same char
  c.compareTo(c1)  0 if c == c1, < 0 if c < c1, > 0 if c > c1.
  c.toString()  c's wrapped char, as a String

Class String

String s = "CS211";

String special place in Java: no need for a new-expression. String literal creates object.

Find out about methods of class String:
docs.oracle.com/javase/7/docs/api/index.html/java/lang/String.html

Lots of methods. We explain basic ones

Important: String object is immutable: can't change its value. All operations/functions create a new String object.
Operator +

"abc" + "12$" evaluates to "abc12$"

If one operand of concatenation is a String and the other isn’t, the other is converted to a String.

Sequence of + done left to right

(1 + 2) + "ab$" evaluates to "3ab$"

"ab$" + 1 + 2 evaluates to "ab$12"

Operator +

System.out.println("c is: "+ c + ", d is: "+ d + " , e is: "+ e);

Can use + to advantage in println statement. Good debugging tool.

Output:

c is: 32, d is: -3, e is: 201

Other useful String functions

s.trim() – s but with leading/trailing whitespace removed
s.indexOf(s1) – position of first occurrence of s1 in s (-1 if none)

s.lastIndexOf(s1) – similar to s.indexOf(s1)

s.contains(s1) – true iff String s1 is contained in s2

s.startsWith(s1) – true iff s starts with String s1

s.endsWith(s1) – true iff s ends with String s1

s.compareTo(s1) – 0 if s and s1 contain the same string,
< 0 if s is less (dictionary order),
> 0 if s is greater (dictionary order)

There are more functions! Look at the API specs!

Window Run Configurations

This Arguments pane of Run Configurations window gives argument array of size 3:

args[0]: "SpeciesData/a0.dat"
args[1]: "2"
args[2]: "what for?"

Click Arguments pane

Quotes OK, but not needed because of space char

Quotes needed

Giving method main an argument

public static void main(String[] args) {

In Eclipse, when you do menu item
Run -> Run or Run -> Debug
Eclipse calls method main. Default is main(null);

To tell Eclipse what array of Strings to give as the argument, Use menu item
Run -> Run Configurations...
or
Run -> Debug Configuration...

(see next slide)

Picking out pieces of a String

s.length(): number of chars in s — 5

01234 Numbering chars: first one in position 0

s.charAt(i): char at position i

s.substring(i): new String containing chars at positions from i to end — s.substring(2) is ’13’

s.substring(i, j): new String containing chars at positions i..(j-1) — s.substring(2,4) is ’13’

Be careful: Char at j not included!

s trim

Be careful: Char at j not included!

String@x2

Length!

CharAt(int)

Substring(int, int)

… more …

… more …

Using several lines increases readability

Operator +

System.out.println("c is: " + c + " , d is: " + d + " , e is: " + e);

Using several lines increases readability

Can use + to advantage in println statement. Good debugging tool.

• Note how each output number is annotated to know what it is.

Output:

c is: 32, d is: -3, e is: 201

19

Operator +

System.out.println("c is: " + c + ", d is: "+ d + ", e is: "+ e);

Using several lines increases readability

Can use + to advantage in println statement. Good debugging tool.

• Note how each output number is annotated to know what it is.

Output:

c is: 32,

19