Characters, Strings, Reading Files

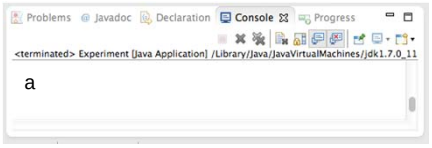
CS2110, Week 2 Recitation

Primitive type char

Use single quotes

Unicode: 2-byte representation
Visit www.unicode.org/charts/
to see all unicode chars

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



Special **chars** worth knowing about

- ' ' - space
- '\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 - On Windows, in CMD and Notepad, to get to a new line manually, you need "\r\n"

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 65

(**int**) 'a' gives 97

(**char**) 97 gives 'a'

(**char**) 2384 gives 'ॐ'

Om, or Aum, the sound of the universe (Hindi)

No operations on **char**s (values of type char)! BUT, if used in relation or arith, a **char** is automatically cast to type **int**.
Relations < > <= >= == != ==

'a' < 'b' same as 97 < 98, i.e. true

'a' < 'Z' same as 97 < 90, i.e. false

'a' + 1 gives 98

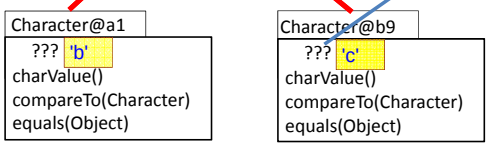
Class Character

An object of class Character wraps a single **char** (has a field that contains the **char**)

```
Character c1= new Character('b');
Character c2= new Character('c');
```

Don't know field name

c1 Character@a1 c2 Character@b9



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
…	…

Static methods in class Character

Lots of static functions. You have to look to see what is available. Below are examples

isAlphabetic(c)
isDigit(c)
isLetter(c)
isLowerCase(c)
isUpperCase(c)
isWhitespace(c)

These return the obvious boolean value for parameter c, a **char**

Whitespace chars are the space ' ', tab char, line feed, carriage return, etc.

toLowerCase(c)
toUpperCase(c)

These return a char.

7

== versus equals

c1 == c2 false
c3 == c1 false
c1 == c1 true

true iff c1, c2 contain same values

c1.equals(c2) true
c3.equals(c1) Error!!!

true iff c2 is also a Character object and wraps same char as c1

c1 Character@a1 c2 Character@b9 c3 null

Character@a1
??? 'b'
charValue()
compareTo(Character)
equals(Object)

Character@b9
??? 'b'
charValue()
compareTo(Character)
equals(Object)

8

Class String

String s = "CS211";

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

s String@x2

String@x2
??? "CS211"
length()
charAt(int)
subString(int)
subString(int, int)
equals(Object)
trim()
contains(String)
indexOf(String)
startsWith(String)
endsWith(String)
... more ...

Find out about method 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 new String objects

9

Operator +

Catenation, or concatenation

+ is overloaded

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

If one operand of catenation 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"

10

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

c 32 d -3 e 201

11

Picking out pieces of a String

s.length(): number of chars in s — 5

01234
"CS 13"

Numbering chars: first one in position 0

s.char(i): char at position i of s — s.charAt(3) is '1'

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!

String@x2
? "CS 13"
length()
charAt(int)
subString(int)
subString(int, int)
... more ...

s String@x2

12

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!

Reading files

On the first assignment, your program will read a file from your hard drive. We will give you the code to hook up to a file and show you what method to read the next line, but some intro to I/O will help.

I/O classes are in package java.io.
To import the classes so you can use them, put

```
import java.io.*;
```

Before the class declaration

Input Streams

Stream: a sequence of data values that is processed —either read or written— from beginning to end. We are dealing with input streams.

Read input stream for a file is by creating an instance of class FileReader:

```
FileReader fr= new FileReader(arg that describes file to read);
```

```
fr.read()        // get next char of file
```

Too low-level! Don't want to do char by char.

Reading a line at a time

Class BufferedReader, given a FileReader object, provides a method for reading one line at a time.

```
FileReader fr= new FileReader(arg that describes file to read);
BufferedReader br= new BufferedReader(fr);
```

Then:

```
String s= br.readLine(); // Store next line of file in s
```

When finished with reading a file, it is best to close it!

```
br.close();
```


Example: counting lines in a file

```
/** Return number of lines in f.
    Throw IO Exception if problems encountered when reading */
public static void printSize(String f) throws IOException {
    FileReader fr= new FileReader(f);
    BufferedReader br= new BufferedReader(fr);
    int n= 0;  // number of lines read so far
    String line= br.readLine();
    while (line != null) {
        n= n+1;
        line= br.readLine();
    }
    br.close();
    return n;
}
```

We explain exceptions later. For now, put this in or Java complains

Don't forget!

(write as while loop or for loop)

Always use this structure!

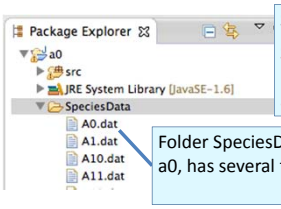
```
line= first line;
while (line != null) {
    Process line;
    line= next line;
}
```


FileReader(String)

When calling FileReader with a String argument s, s can be a name relative to the Eclipse project you are running.

When running a procedure main in Project a0, because folder SpeciesData is in a0, to read file A0.dat, we can use

```
FileReader fr= new FileReader("SpeciesData/A0.dat");
```



Separate names in path with /.

Slides 20-21: don't have to hardcode file name, can give it as arg in call to main

Folder SpeciesData, in project a0, has several files in it

FileReader(File)

An object of class File contains the path name to a file or directory. Class File has lots of methods, e.g.

f.isDirectory() --return true if f is a directory
f.length() --length of file f
f.list() -- return a String array of names of files and directories in directory f

So we often work with a File object instead of a String

```
File f= new File("SpeciesData/A0.dat");
FileReader fr= new FileReader(f);
```

19

Given method main an argument

```
public static void main(String[] args) { … }
```

Parameter: String array

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)

20

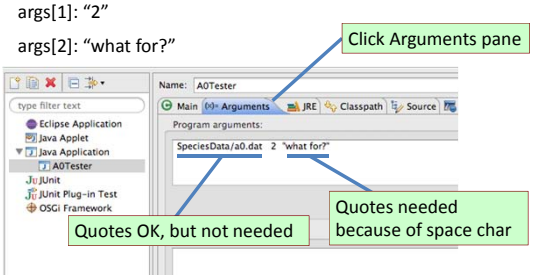
Window Run Configuations

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?"



21