

News

Room	Feb 5	Mar 12
HA403 (Tuesday Lab Students)	Ali Reza Sadeghian Dion Lew	Ali Reza Sadeghian Alagan Apalagan
GB405 (Monday Lab Students)	Paul Hellyar Jianghong Hu Me (I may go to HA403 too)	Paul Hellyar ??anyone?? Me

- quiz:
 - 50 minutes
 - 19 questions (5 marks each... last one 10 marks)
 - some really easy, some difficult, all do-able
 - some people will find it long
 - *goal*: maximize your marks
 - *best strategy*: answer easiest questions first
 - leave hard questions till end
 - *everybody* should pass
 - difficult to get 100% unless you *really* know your stuff
 - mark should be indicator of “how much you know”
 - mark tells you “how much to learn” over reading week

Last Day

- objects: comparing, assignment
- public, private

Today

- “this”
- Strings (finally!)

Strings are Objects (Text 2.6)

Strings so far are in quotes: “Hello, world!”

- part in quotes is called a *string literal*
- we can store in a “variable” as a *String* object

String class defined by Java

- just another class... just another variable
- many built-in methods:
 - `length()`, `substring()`, `toLowerCase()`...
- but Java treats Strings as *special* objects, lets you use +

Example

```
class StringTest
{
    public static void main(String[] args)
    {
        String h = "Hello, world!";
        //           111
        //           0123456789012

        String g;
        g = h.substring(1, 5);
        // h.substring( start, pastEnd );

        System.out.println( g + h );
    }
}
```

prints:

```
elloHello, world!
```

Public and Private

- in Lab 4, make numerator and denominator **private**

Using “this”

- what if we name parameters “real” and “imag”?

```
class Complex
{
    double real, imag;

    public Complex( double real,
                  double i )
    {
        boolean imag = true;
        this.real = real;
        this.imag = i;
    }
    ...
}
```

problem

- param “real” overrides instance variable “real”
- variable “imag” overrides instance variable “imag”
- “real” is just the parameter “real”
- “imag” is the local (boolean) variable

solution

- word “this” refers to “this object”...
 - use to access instance variables and methods of object
- “`this.real`” forces it to be the instance variable `real`
- “`this.imag`” forces it to be the instance variable `imag`

note: static methods have no “this” !!!!!

Special Characters (in *String* and *char*)

Suppose you want to print

```
Hello
World
```

with only one `println()` statement...

```
System.out.println("Hello\nWorld");
```

The `\n` is a special way to write “I want a new line”

- only takes the space of *one* character
- you write it inside a string as 2 characters: `\` and `n`
- javac sees “`\`” ... it treats the next character in a special way:
 - `\n` means “a newline character”
 - `\t` means “a tab character”
 - `\r` means “a carriage return character” (no new line)
 - `\"` means a “ character

But how do I print just a “`\`” ?

- `\\` means a “`\`” character

Examples

```
"Hello\nworld"    =>  Hello
                    world
"Hello\tworld"   =>  Hello world
"\"Hello world\"" =>  "Hello world"
"\\Hello world/" =>  \Hello world/
"Hello world\rJ" =>  Jello world
```

When using the *char* type, can also use `\n`

```
char newline = '\n';
char tabchar = '\t';
char dbl_quote = '\"';
char sgl_quote = '\'';
```

Strings as Objects

- can pass strings as parameters
- can return strings as results

```
class Complex {
    ... other methods ...

    public String toString() {
        return "this is a complex number.";
    }

    public static Complex
    getComplex(String prompt)
    {
        System.out.println(prompt);
        System.out.print("real part:");
        double r = Stdin.getDouble();
        System.out.print("imag part:");
        double i = Stdin.getDouble();
        return new Complex(r,i);
    }
}

class ComplexTest
{
    public static void main(String[] args)
    {
        Complex cnum;

        cnum= getComplex("Enter complex number");
        String str = cnum.toString();

        System.out.println( str );
        System.out.println( cnum ); //magic!
    }
}
```

A Few String Methods...

```
int length()
• returns length of string (number of chars)
• h.length() == 13

String substring(int start, int pastEnd)
String substring(int start)
• returns a smaller part of a given string, beginning at position
  start and up to (but not including) pastEnd
• if pastEnd omitted, goes to end of string
• h.substring(2) returns "llo, world!"
• h.substring(2,5) returns "llo"
• why pastEnd? because (pastEnd - start) = length

String toLowerCase()
String toUpperCase()
• returns an all-lowercase (or all-upercase) version
• h.toLowerCase() returns "hello, world!"
• h.toUpperCase() returns "HELLO, WORLD!"

String trim()
• returns a trimmed string where all whitespace (spaces, tabs,
  newlines) from beginning and end of string are removed
• h.substring(0,7) returns "Hello, "
• h.substring(0,7).trim() returns "Hello,"

char charAt(int position)
• returns a single char from the string at position
• h.charAt(7) returns 'w'
• char is a 16-bit Unicode value
• 'A' = 65, 'B' = 66, ... 'a' = 97, 'b' = 98, ...
• Unicode is extension of ASCII; supports many foreign language
  characters (up to 65,535 characters)
• ASCII: American Std. Code for Information Interchange
```

String Concatenation

- + is used to *concatenate* strings to make a longer one
- if either side of + is a string, Java *automagically* tries to *convert* other side to a string
- if other side is ANY TYPE of object, it *automagically* *invokes method* called toString() to convert to String **STRING MAGIC!!!**

Examples

```
int x=3,y=4;
Complex cnum = new Complex(3.0,-1.2);
System.out.println("x = "+x+ "+" +y );

prints:
x =3+4

System.out.println( ... ) => prints
"x"+"y" => "xy"
x+y => "7"
"x"+y => "x4"
x+"y" => "3y"
"x"+x+y => "x34"
"x"+x*y => "x12"
x+y+"y" => "7y"

STRING MAGIC!!! cnum => "3.0+(-1.2)*i"
"A"+cnum => "A3.0+(-1.2)*i"
cnum+x => error!!!
x+""+cnum => "33.0+(-1.2)*i"
```

```
class Complex {
    ...
    public String toString() {
        return real + "+" + imag + "i";
    }
    ...
}
```

Comparing Strings

Question: Which string comes first?

- "g" or "G" ?
- " " or "g" ?
- " g" or "g " ?
- "goodbye" or "good" ?
- "goodbye" or "hello" ?
- ordering in computers:
 - " " < "G" < "H" < "g" < "h"
 - "g" < "g " < "ga"

Answer: Depends on who you are?

- are you the phone company?
 - "AA Towing" comes before "A A Towing"
- are you a computer?
 - compare character-by-character
 - use Unicode (or ASCII) order of individual characters
 - if characters differ
 - string with smaller char (Unicode value) comes first
 - "A A Towing" comes before "AA Towing"
 - if characters all same, but one string is shorter
 - smaller string comes first
 - "good" comes before "goodbye"

```
int compareTo(String other)
• compares to other string in lexicographic (dictionary) order
• returns <0 if other comes after (this < other)
• returns 0 if other equals (this == other)
• returns >0 if other comes before (this > other)
• h.compareTo("Hello, world!") returns 0
• h.compareTo("Goodbye") returns >0
• h.compareTo("goodbye") returns <0
```

Immutable Strings

- String objects are *immutable*!!!!
 - reference to string object will *ALWAYS* point to same string
 - all string methods return a new string object
 - program cannot “change” the value of the string object
 - only need 1 string object for each unique string

```
class Test
{
    public static void main(String[] args)
    {
        String s1 = "hello";
        String s2 = "world";

        System.out.println(s1 + s2);
        s1 = s2; // s1,s2 point to same object
        s2.substring(2,5); // doesn't use result
        s2 = s2.substring(2,5);
        s2 = "ruler"; // s1 still "world"
        System.out.println(s1 + s2);
        if( s1 == s2 ) // shouldn't do this
        if( s1.equals(s2) ) // do this!
            System.out.println("equal");
    }
}

// prints:
//helloworld
//worldruler
```

- immutable is an important object concept
- thinking problem...
 - how would you make class Rational immutable?
 -