

ITEC1620
Object-Based Programming

Lecture 19
Strings

Text

- Text primitive is char – a single character
- Useful text (e.g. words) are multiple characters
- What do we use for multiple variables of the same datatype?
 - Arrays
- Arrays of characters are Strings

Strings

- JAVA comes with the String class
 - Fields
 - Array of characters
 - Methods
 - Useful manipulations
 - substring(), toUpperCase(), replace(), etc

Strings II

- Strings / character arrays pre-date JAVA
 - Primitive datatype in some languages
- Three important (primitive) operations exist
 - Initialization
 - Concatenation
 - Conversion

Initialization

- Anything inside double quotes is a string literal
- String literals appear as a primitive datatype
 - `int aNumber = 5;`
 - `String aString = "Hello";`

Initialization II

- String literals are not normal JAVA syntax – hides that an object is created
- Normal JAVA syntax

```
char[] text = new char[] {'H', 'e', 'l', 'l', 'o'};  
String aString = new String(text);
```

Concatenation

- Strings can be “added” together

```
String twoWords = “first” + “second”;
```

- Normal JAVA syntax

```
String first = “first”;
```

```
String second = “second”;
```

```
String twoWords = first.concat(second);
```

Conversion

- Primitive datatypes can be converted into Strings through concatenation

```
String aNumber = 5;      //syntax error
```

```
String aNumber = "" + 5;
```

- Normal JAVA syntax

```
Integer num = new Integer(5);
```

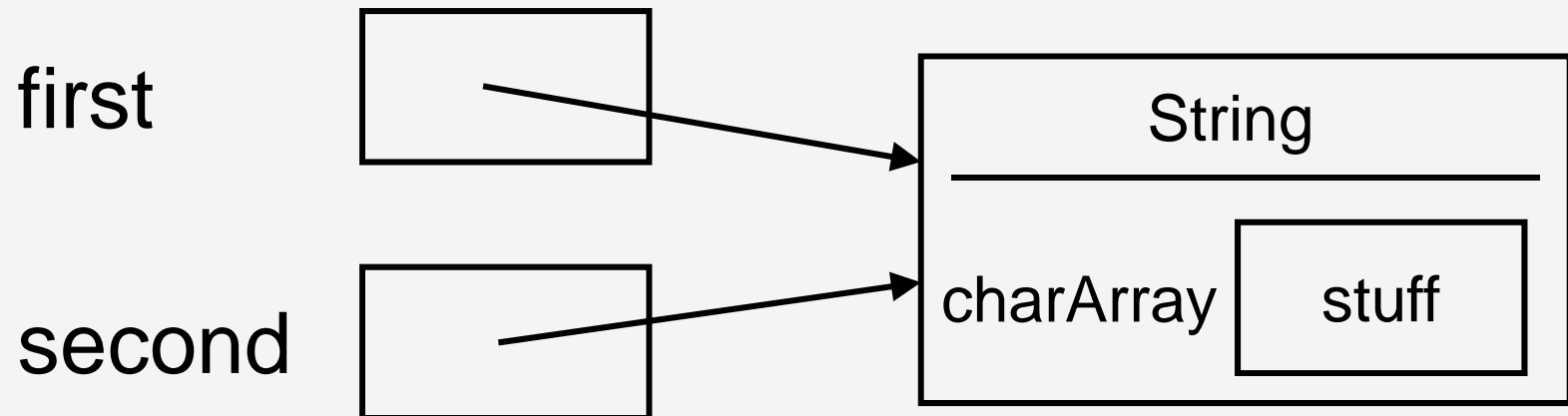
```
String aNumber = num.toString();
```


Inconsistency

- Strings look like primitive datatypes
String first = "stuff";
String second = "stuff";
- Should first == second?
 - Yes – if Strings are primitive datatypes
 - No – if Strings are normal objects

Inconsistency II

- To make `first == second` true, JAVA makes the identifiers refer to the same object



Inconsistency III

- What happens if second is changed?
 `second.toUpperCase();`
- If the String referred to by second is changed, then the String referred to by first will also change
 - The independent declarations do not imply this – cannot let it happen

Inconsistency IV

- Solution – make strings “immutable”
 - All methods that “change” Strings return a new String object

```
String twoWords = first.concat(second);
```

```
System.out.println(first);
```

```
//first
```

```
System.out.println(twoWords);
```

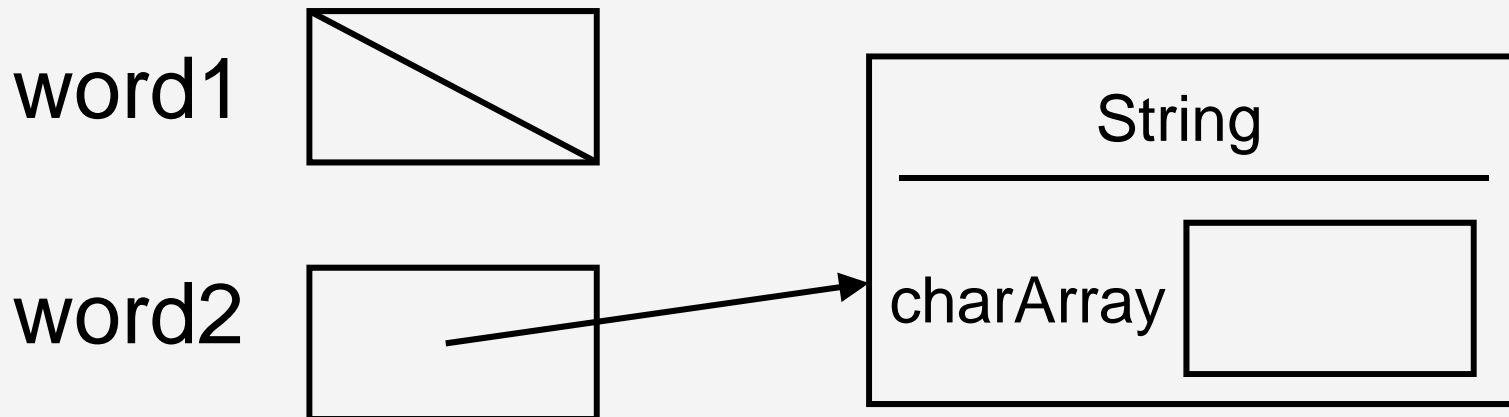
```
//firstsecond
```

Null String and Null object

String word1 = null;

String word2 = "";

- The null String is not the null object



Questions?

Object-Based Programming

- Write a program that calculates the sum of the digits for an input positive int
 - Do not use divide or modulus
 - Use methods from the String and Integer classes

```
public static void main (String[] args)
```

```
{
```

```
}
```


Arrays Programming

- Write a program in Java that implements String concatenation.

```
//String first;  
//String second;  
//String sum;
```

Problems with Arrays

- Must know their length before allocating them
- Want dynamic structures (ITEC2620)
- Don't want to deal with them yet...
 - Put arrays (or other lists) inside of an object
- Java has collections

Examples

Warehouse

// Money totalValue

// Warehouse warehous

File System

// MemoryCard card

Readings and Assignments

- Text sections (5th or 6th edition)
 - 2.1, 3.2, 7.7, 12.1
- Text sections (7th edition)
 - 2.1, 3.2, 5.6, 13.1
- Strings Program (Tutorial)