Strings

Strings and their methods

Produced by: Dr. Siobhán Drohan
Mr. Colm Dunphy
Mr. Diarmuid O'Connor

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
   - **Internal**
   - **External**
   - **Dot notation**
7. Using String methods: some **examples**

Recap: Primitive Types

- Java programming language supports eight primitive data types.
- The char data type stores one single character which is delimited by single quotes(‘)
  e.g.
  ```
  char letter = ‘a’;
  ```

Data Type	Default Value
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
boolean	false

Primitive Types: **char**

```
// VALID USE
char letter  = ‘n’;         //Assign ‘n’ to the letter variable
char letter = ‘N’;          //Assign ‘N’ to the letter variable

// INVALID USE
char letter = n;            //ERROR – no single quotes around n.
char letter = “n”;          //ERROR – double quotes around n.
char letter = “not”;        //ERROR – char can only hold one character.
```

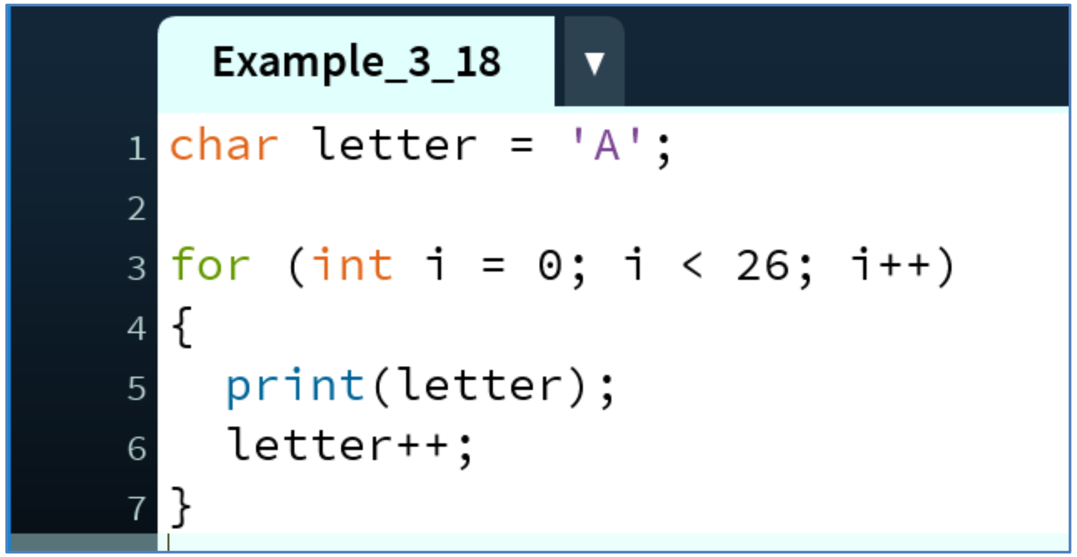
Source: Reas & Fry (2014)

Primitive Types: char

- char is a 16-bit Unicode character.
- It’s values range:
  - from '\u0000‘ (or 0)
  - to '\uffff' (or 65,535)
- For example:
  - ‘A’ is ‘\u0041’
  - ‘a’ is ‘\u0061’

http://en.wikipedia.org/wiki/List_of_Unicode_characters

Example 3.18 – Alphabet



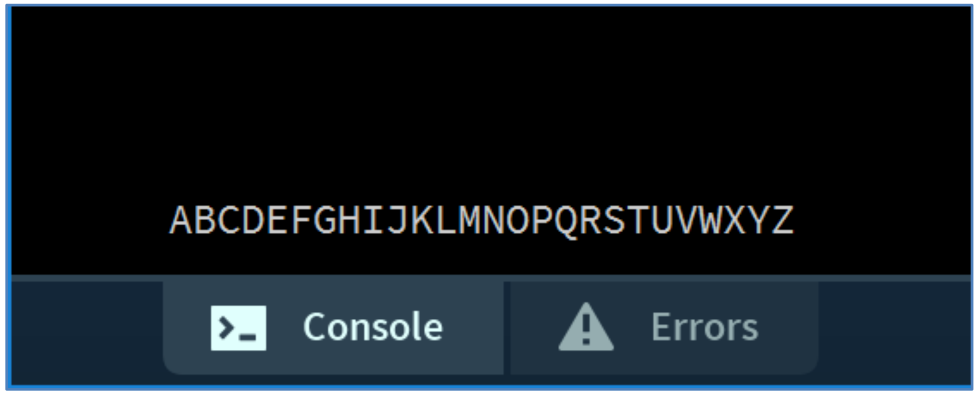
Example_3_18

```
char letter = 'A';

for (int i = 0; i < 26; i++)
{
  print(letter);
  letter++;
}
```

This code uses the underlying **Unicode** value for ‘A’ (i.e. ‘\u0041’)
and adds one to it each time the for loop is iterated.

As the for loop is iterated 26 times,
and the starting value is ‘A’,
our loop will print the alphabet to the console.



```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Source: Reas & Fry (2014)

Topics list - **Strings**

Object types e.g. String

- Strings, which are widely used in Java programming, are a sequence of characters enclosed by double quotes (“”).
  e.g. **“seq of chars”**

- In Java, a **String** is an **object type**.
- The Java platform provides the **String class** to create and manipulate strings.
- The most direct way to create a **String** is to write:

```
String greeting = "Hello world!";
```

http://www.tutorialspoint.com/java/pdf/java_strings.pdf

Object types - **String**

```
// VALID USE
String str = “I am a sentence”;  //Assigns the full sentence to str variable.
String word = “dog”;       //Assigns the word “dog” to the word variable.
String letter = “A”;       //Assigns the letter “A” to the letter variable.

// INVALID USE
String letter = n;         //ERROR – no double quotes around n.
String letter = ‘n’;       //ERROR – single quotes around n; use double.
string letter = “n”;       //ERROR – String should have a capital S.
```

Object Data Types start with a Capital Letter to distinguish them from Primitive data types

Source: Reas & Fry (2014)

Topics list - **Strings**

1. Primitive Types: **char**
2. Object Types: **String**
3. **Primitive** Types **versus Object** Types
4. Strings and **Java API**
5. Strings - **methods**
6. **Method calls**
   - **Internal**
   - **External**
   - **Dot notation**
7. Using String methods: some **examples**

Primitive types vs. **Object** types

Primitive type

```
int i = 17;
```

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Directly stored in memory...

$\boxed{17}$

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

Directly stored in memory…

$\boxed{17}$

Object type

```
String hi = "Hello";
```

Primitive types vs. Object types

Primitive type

```
int i = 17;
```

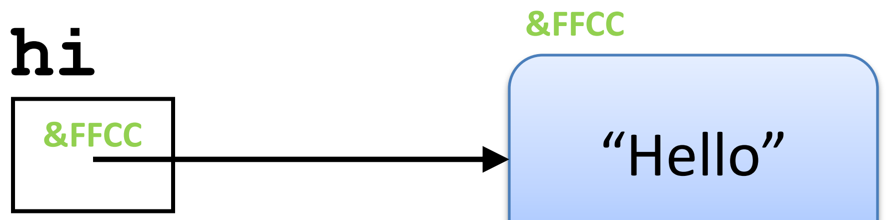
Directly stored in memory…

$\boxed{17}$

Object type

```
String hi = “Hello”;
```

hi variable
contains a reference (*address*) to where the String is stored in memory



Primitive types vs. Object types

Primitive type

```
int i = 17;
```

With **primitive type** variables (e.g. int, float, char, etc)

Directly stored in memory…

$\boxed{17}$

the **value** of the variable is stored in the memory location assigned to the variable.

Primitive types vs. **Object** types

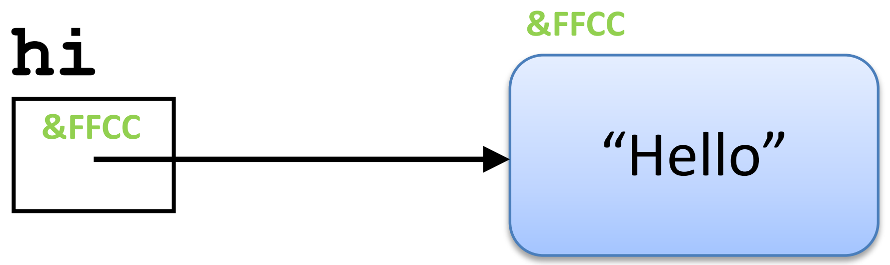
With object types,
the variable holds the **memory address**
of where the object is located
– not the values inside the object.

This memory address is called
a **reference** to the object.

Object type

```
String hi = "Hello";
```

hi variable
contains a reference (*address*)
to where the String is stored in memory



Primitive types vs. Object types

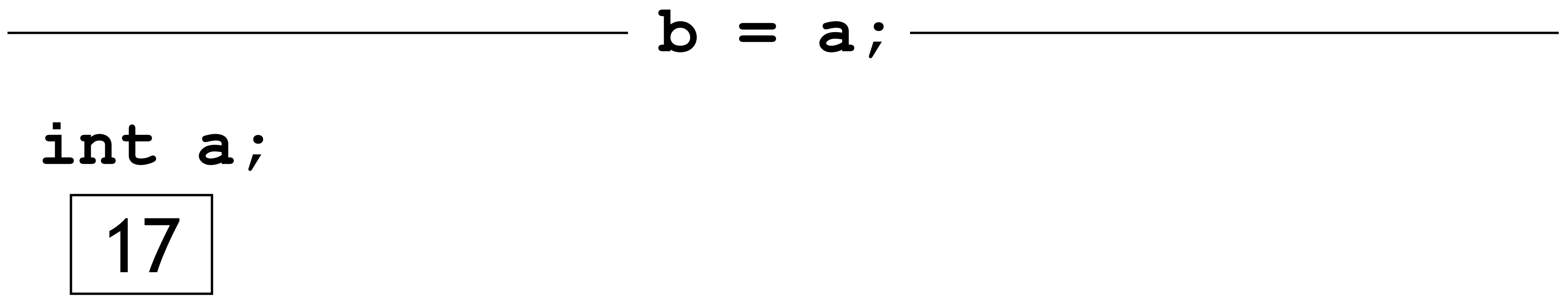
Now that we know how primitive types and object types store data,

we will look at this statement (b=a) in the context of primitive and object types.

```
b = a;
```

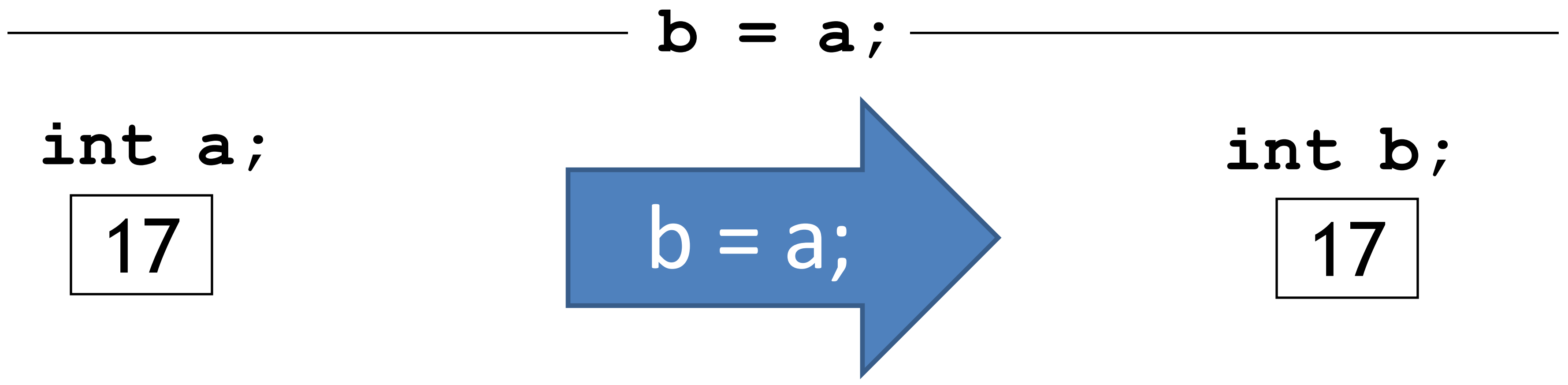
Primitive types vs. Object types

Primitive types



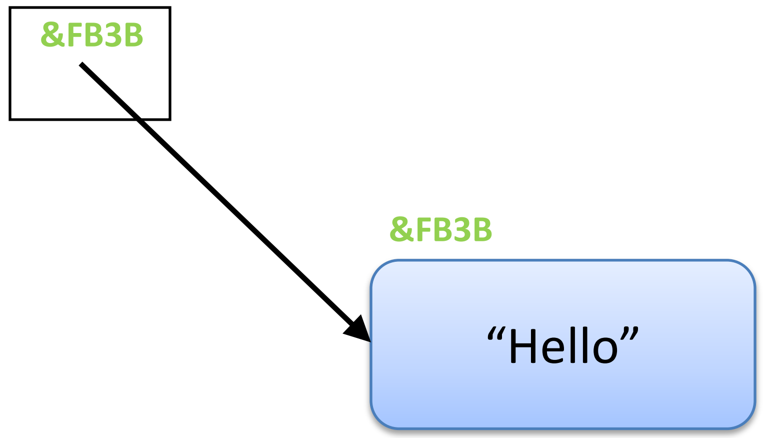
Primitive types vs. Object types

Primitive types



Primitive types vs. **Object** types

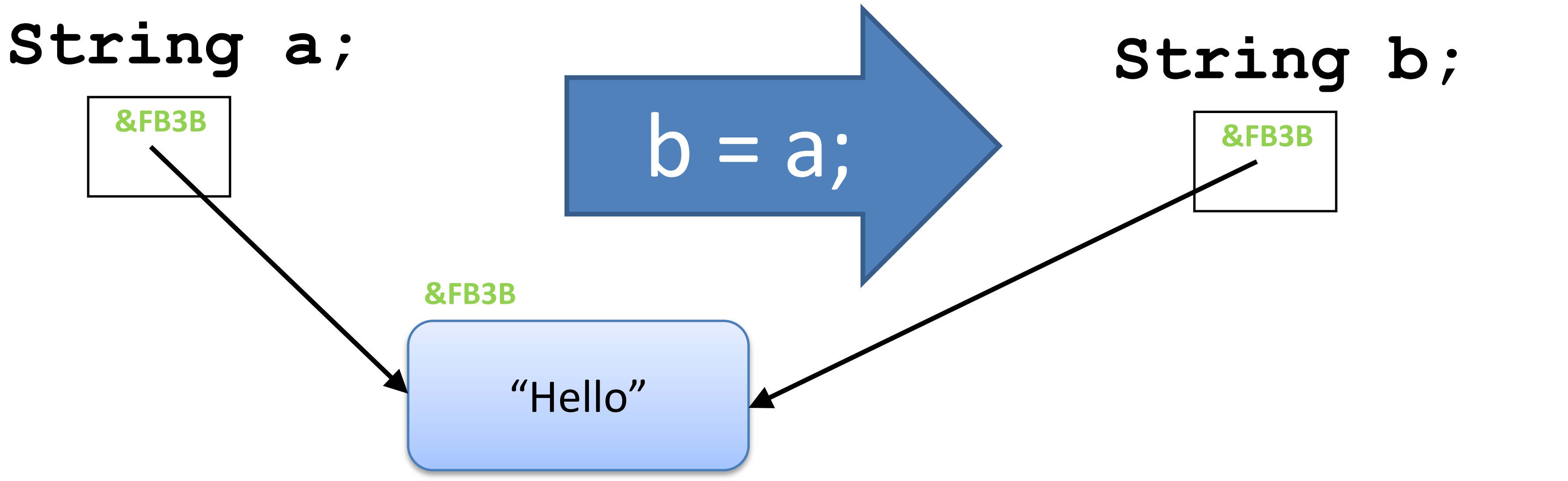
```
String a;
```



```
b = a;
```

Object types

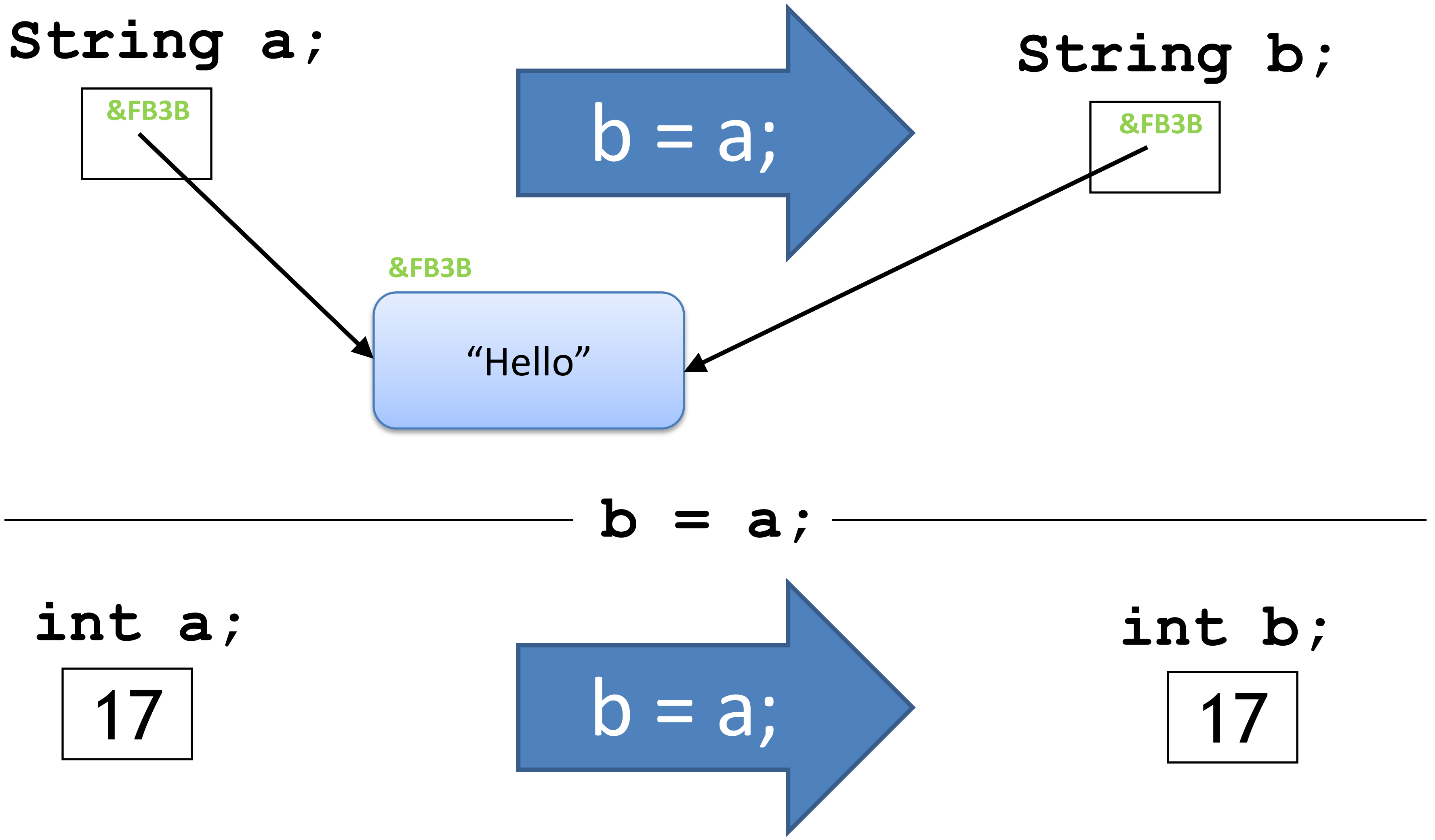
Primitive types vs. **Object** types



```
b = a;
```

Object types

Primitive types **vs.** Object types



Questions?

References

- Reas, C. & Fry, B. (2014) Processing – A Programming Handbook for Visual Designers and Artists, 2nd Edition, MIT Press, London.