# CSC 142

References and Primitives
[Reading chapter 5]



## Review: references and primitives

- Reference: the name of an object. The type of the object is defined by a class, e.g.

```
Car myCar = new Car("sedan");
//myCar is a reference to an object of
//type Car
```

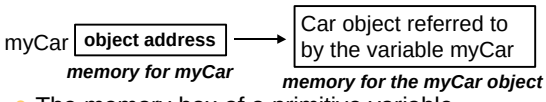
- Primitive: a variable to store a simple piece of information, such as an integer, a floating point number, a character, or a boolean. The type is defined by one of the java keywords, int, double, char, boolean, etc…

```
int i = 10; // don't use new
```

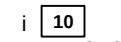


## Memory view

- All program variables are stored in memory. Think of memory as a huge pile of boxes. Each box is known to the computer by its location (=address).
- The memory box of a reference variable contains the memory address where the object that the variable refers to is stored.




- The memory box of a primitive variable contains the value of the variable






## Method call

- When calling a method, the values of the actual parameters are copied in the formal parameters. We say that java calls by value.
- If the actual parameter is
  - a primitive: the value of the primitive is copied
  - a reference: the value of the reference is copied (=address of the object). Actual and formal parameters refer to the same object.



## Example

```
// foo method
public void foo(StringBuffer sb, int j){
  sb.reverse();//reverse the order
   j = 2*j;
}

// in some method bar calling foo
int i = 10;
StringBuffer s = new StringBuffer("Hello");
foo(s,i);
System.out.println("s="+s+" i="+i);
```

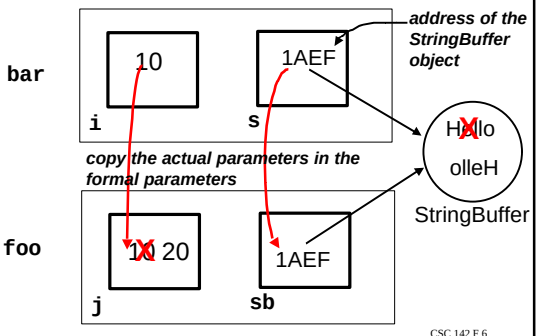
*Like the String class. But a StringBuffer comes with methods to modify its content (the content of a String can't be modified. String are immutable).*

- What is printed?

```
s=olleH i=10
```



## What is going on?



## Using references and primitives

- An object can be huge in memory
  - passing an object to a method by copying the object would be prohibitively expensive
- A primitive never uses lots of memory. It can be passed as a copy.
- Note: a primitive in one method can't be modified by another method when passed as an actual parameter to that other method.
  - But sometimes we want to do so!
  - Answer: store the primitive inside of an object (e.g. an array) and call the method with a reference to that object as an actual parameter.