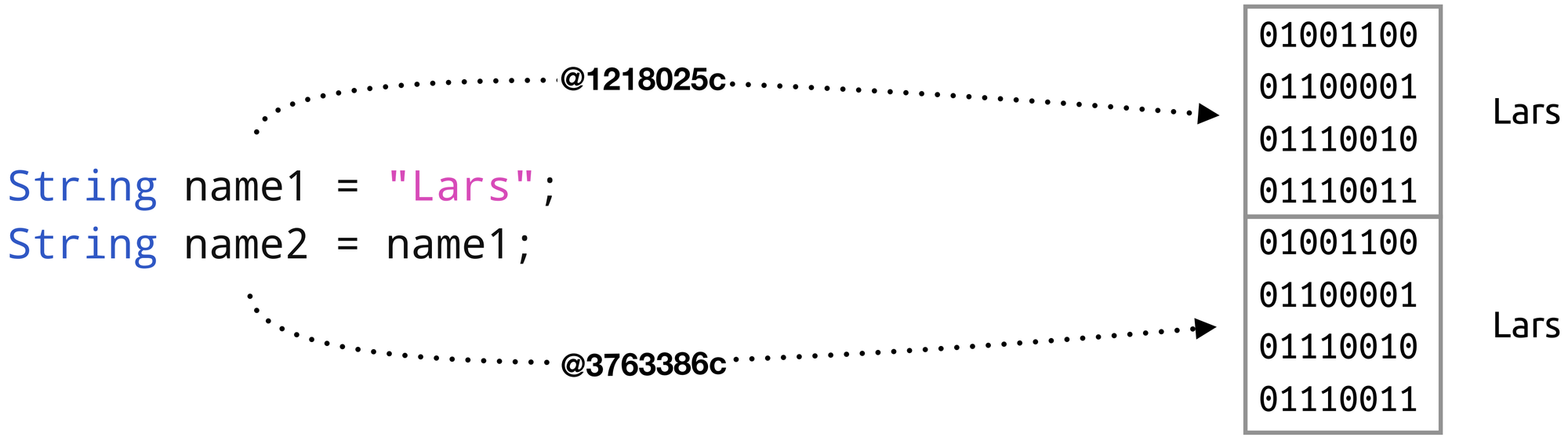
Essential Computing 1

Strings & characters

Recall that **Strings behave like a primitive types**.
(that is; they behave like *value types*)



But really, **Strings are objects**, and they contain methods.

```
String name = "Lars";
boolean isFars = name.equals( "Fars" ); // false
```

Some **useful methods**

```
String name = "Lars";
boolean isFars = name.equals( "Fars" ); // false
int characterCount = name.length(); // 4
String namePart = name.substring( 2, 2 ); // "rs"
char letter = name.charAt( 1 ); // 'a'
int aIndex = name.indexOf( 'a' ); // 1
String upperName = name.toUpperCase(); // "LARS"
String upperName = name.toLowerCase(); // "lars"
String changedName = name.replace( "L", "F" ); // "Fars"
String letterDifference = name.compareTo( changedName ); // 1
String splitName = name.split( "r" ); // [ La, s ]
```

Strings are **immutable** (unchangeable).
Any string manipulation creates a new String object.

```
String name = "Lars";

name.toUpperCase(); // Name is not changed by this.

// The 'toUpperCase' method returns a new String object that
// we need to assign to the variable.
name = name.toUpperCase();

// The addition operator also creates a new String object.
name = name + ".";
```

Strings contain characters. Get them using charAt.

```
String name = "Lars";
String stretchedName = "";
for( int i=0; i<name.length(); i++ ){
    char letter = name.charAt(i);
    stretchedName += letter + "  ";
}

// stretchedName will contain "L  a  r  s" at this point.
```

Characters are encoded as **unicode**

æøå ... like in Danish æblegrød and selvmål.

 ...looks like an alien "reddit" guy.

 ...squid?

◉ ▣ ■ ▫ ☐ ◯ ▲ ▶ ▶ ▼ ◆ ◢ ◣ ◤ ◥ ... geometric shapes

𝒻𝒻 ⋰ ⋱ ∶ ∷ ∴ ∵ ⋘ ⋙ ... math!

https://unicode-table.com

The unicode number is often expressed as hexadecimal

```
char eternitySign = 0x058E;
```

Unicode number: U+058E

HTML-code: ֎