The Parts of a Java Program’s Memory
A Running Program’s memory is a finite expanse of consecutive addressable storage locations, each is 8 bits (1 byte) wide.

Address goes from 0 to some max.

A program’s code and data are stored in its memory.

- When you declare a variable, a memory location is allocated for it at some address, the variable’s name is how you refer to the memory location in your program:

```c
int x; // allocate memory for an int, and associate the name “x” with it
x = 10; // store the value 10 at the memory location associated with “x”
```

<table>
<thead>
<tr>
<th>Address:</th>
<th>variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>how my program refers to memory locations (aliases for addresses)</td>
</tr>
<tr>
<td>1:</td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>10</td>
</tr>
<tr>
<td>…</td>
<td></td>
</tr>
<tr>
<td>max:</td>
<td>: x</td>
</tr>
</tbody>
</table>
Parts of a Java Program’s memory

- Variables are allocated in one part of the running program’s memory (called the Stack)

- Objects are allocated in a different part of the running program’s memory

Strings

```java
String str1, str2;
int x;

str1 = new String("testing");
x = str1.length();
str2 = new String("hello");
```
• Static data and methods of a class are in another part of memory

Class Foo {
    public final static int MIN = 10;
    public static int FooCount() {...
    public static void main(...

    Class Foo’s static parts:
    MIN 10
    FooCount method: ...
    main method: ...

    Stack
    str1
    str2
    x 7

    Object Memory
    ```testing''
    length: ....
    ```hello''
    data: ...
    methods: length: ....

    Static Memory
• A Running Program’s memory may look like:

Static Memory: static data and methods go here

Object Memory: objects created with `new` go here

Stack: variables and parameters go here