Java Basics & Syntax

Ignore the memes

Variables

Like with other languages, Java has primitive variables

- `int`
- `double`
- `boolean`
- `char`

Declaring these variables are simple:

```
int age = 5;
double sum = 55.2;
char ogre = 's';
```

Methods

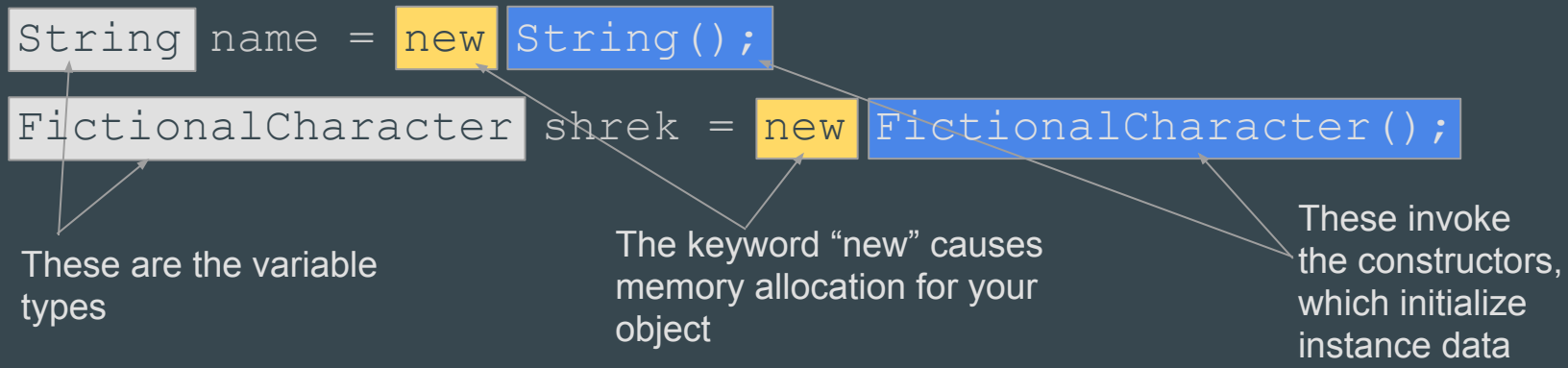
Functions are called methods in Java and are implemented similarly to other languages:

```
public String getName(){
    return name; //name is some predefined variable
}

public int getAge(){
    return age; //age is some predefined variable
}
```

Objects

Java is an object oriented language, meaning that functions and variables are categorized into classes, which can be instantiated as object variables.



Objects (cont.)

Say you want to invoke a method specific to an object. Assume that the class `FictionalCharacter` has a method called `getName()`. To access the `getName()` function from an object of `FictionalCharacter`, you would use the **dot operator**.

```
FictionalCharacter ogre = new FictionalCharacter();
ogre.getName(); //returns the name of the ogre
```

Public, Private, and Protected

When programming a class, many variables and methods will only be used within that class, while others can be used outside of it.

To prevent the end-user from having unauthorized access to critical data/methods, use the keyword **private**.

To allow the user to access methods outside of your class, qualify the declaration with the keyword **public**.

Don't worry about **protected**, you'll rarely (if ever) use it.

Strings

Unlike most languages, Java implements Strings as objects, not primitive data.

```
String donkey = new String(“Eddie Murphy”);
String donkey = “Eddie Murphy”;
```

However, Strings are versatile and can be instantiated and concatenated using conventional primitive methods:

```
donkey += “ voiced me”;
System.out.println(donkey); //displays: Eddie Murphy voiced me
```

Strings (cont.)

Some important String methods to note:

`charAt(int index)`	Returns the char value at the specified index
`contains(CharSequence s)`	Returns true if and only if this string contains the specified sequence of char values.
`equals(Object anObject)`	Compares this string to the specified object.
`indexOf(String str)`	Returns the index within this string of the first occurrence of the specified substring.
`length()`	Returns the length of this string

Arrays

An array is a data structure that holds lists of items. While arrays *are* objects, they are instantiated differently:

```
int[] myList = new int[420];
```

This specifies the type of array (this array holds ints, for example)

This specifies the number of elements in the list - cannot change once declared

Remember that once instantiated, you *cannot* change the size of an array, but you can change the values at each index. Also note that it is not necessary to instantiate the array at the point of declaration.

Arrays (cont.)

Accessing elements in an array is simple, but don't forget that in computer science the first index is always 0 and the last index is one less than the length.

```
myList[0] = 4; //first index
myList[1] = 2; //second index
```

To get the length of the array, use the length field:

```
System.out.println(myList.length); //displays 420
myList[myList.length] = 23; //throws an out of bounds error
myList[myList.length - 1] = 42; //works!
```

Arrays (cont.)

Sometimes, you might need to use a 2D array (think of a list of lists).

```
int[][] twoDList = new int[5][10];
```

Number of columns

Number of rows

This creates a 5x10 2D array of type int.

Arrays (cont.)

To access the number of rows, use the length field

```
twoDList.length; //value of 5
```

To access the number of columns, use the length field of any row [0, length)

```
twoDList[0].length; //value of 10
```

ArrayLists

An ArrayList is like an array but with variable size — you can add or remove elements post construction. However, unlike conventional objects, ArrayLists require more information during construction:

```
ArrayList<String> list = new ArrayList<String>();
```

The `<String>` simply specifies that this is an ArrayList that holds variables of type String.

ArrayList (cont.)

Some important ArrayList methods to note:

`add(E element)`	Appends the specified element to the end of this list.
`get(int index)`	Returns the element at the specified position in this list.
`remove(int index)`	Removes the element at the specified position in this list.
`set(int index, E element)`	Replaces the element at the specified position in this list with the specified element.
`size()`	Returns the number of elements in this list.

Scanners

Many times, you will need to get user input for your program to work properly. In Java, this is accomplished by using a Scanner object.

```
Scanner scanner = new Scanner(System.in);
```

Specifies the input stream to use

Scanners (cont.)

Some important Scanner methods to note:

`next()`	Returns the next complete token from this Scanner
`nextInt()`	Returns the next int from this Scanner
`nextDouble()`	Returns the next Double from this Scanner
`nextLine()`	Returns the next line from this Scanner
`close()`	Closes the Scanner stream

Have any questions?

Don’t ask us, consult the API instead!
Just kidding, but seriously, use the API, it' s a great resource.