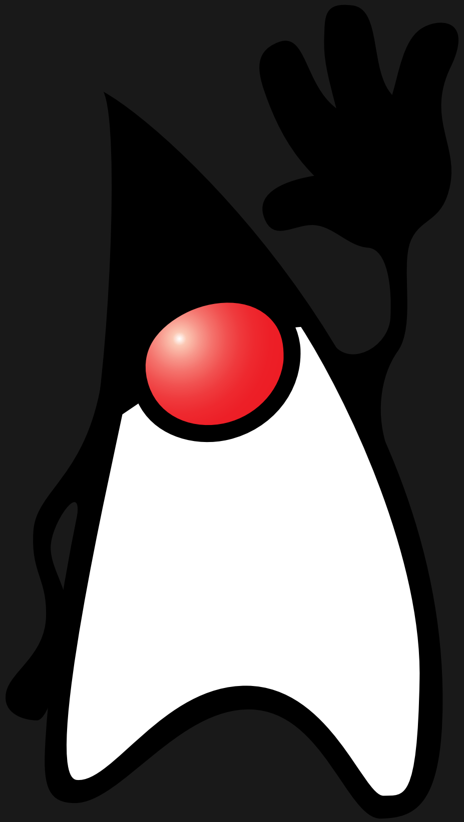
JAVA FOR QA

WEEK 02



AGENDA

- The main Method
- Object-oriented programming
- Java classes && objects
  - methods common to all objects
- Boxed types
- More about String class
- Enum types
- Annotations
- Visibility modifiers

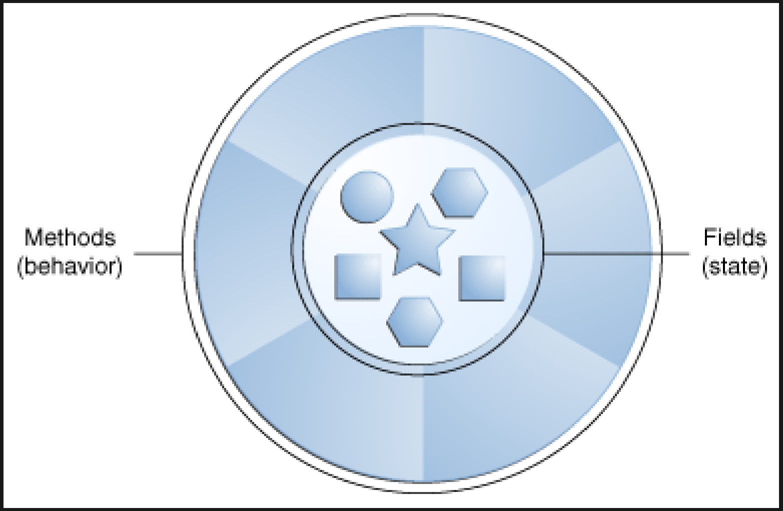
THE MAIN METHOD

```java
/**
 * The HelloWorldApp class implements an application that
 * simply displays "Hello World!" to the standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); //Display the st
    }
}
```

OBJECT ORIENTED PROGRAMMING

- objects
  - state + behavior
- classes
- inheritance
- polymorphism
- encapsulation

OBJECT



BENEFITS

- modularity
- information-hiding
- code re-use
- pluggability and debugging ease

JAVA CLASSES && OBJECTS

- class && interface definition
- creating objects
- inheritance
- fields
- methods
- (default) constructor
- keywords
- this && super

JAVA CLASSES && OBJECTS

- more on fields/variables
  - access control
  - scope
  - initializing
  - naming convention
- nested / innner / anonymous classes

OOP - EXERCISE

- create interface `Shape` with method `getName()` and `getArea()`.
- create some `Shape` implementations: `Circle`, `Rectangle`, `Square`, `Triangle`
- create array with different shapes and for each shape print its name and area

GOOD PRACTICE

Refer to objects by their interfaces.

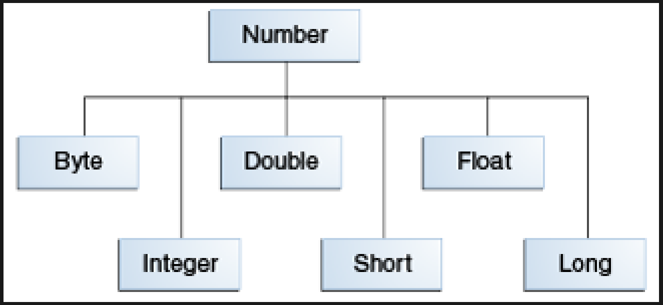
BOXED TYPES

- wrapper classes for primitive types
- autoboxing and unboxing
- use cases
  - when object/class is required (e.g. collections)
  - to use constants, e.g. `Integer.MAX_VALUE`
  - to use class method fir converting from/to other numeric types or string

BOXED TYPES

Primitive type	Wrapper class
boolean	Boolean
byte	Byte
char	Char
float	Float
int	Integer
long	Long
short	Short
double	Double

NUMERIC TYPES



AUTOBOXING

```java
// TODO: What's wrong with this?
/**
 * @return -1 if i < j
 *          0 if i is equal j
 *          1 if i > j
 */
static int compare(Integer i, Integer j) {
  return (i < j) ? -1 : (i == j ? 0 : -1);
}
```

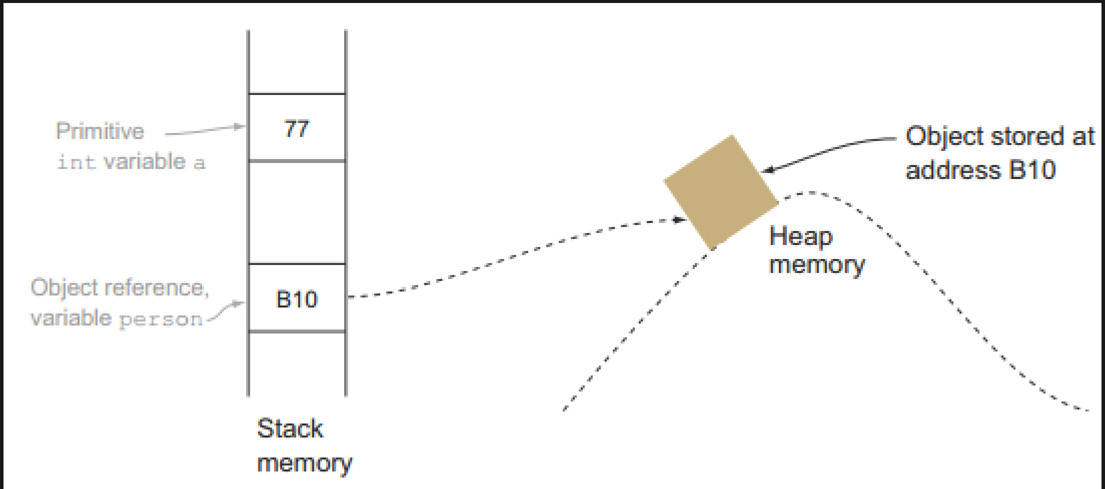


Figure 2.13 **Primitive variables store the actual values, whereas object reference variables store the addresses of the objects they refer to.**

EXERCISE

Calculate sum of all positive integer numbers.