Objective

Using Objects

- To understand the concepts of classes and objects
- To be able to call methods
- To be able to browse the API documentation
- To realize the difference between objects and object references

Study chapter 21 ... 2.8 from your text book (7 edition)

Identifiers

- By convention, variable names start with a lowercase letter
- By convention, class names start with an uppercase letter

```
String greeting = "Hello, World!";
PrintStream printer = System.out;
int luckyNumber = 13;
```

Initialize

- All variables should be initialized before using.

Error:

```
int luckyNumber;
System.out.println(luckyNumber);
   // ERROR - uninitialized variable
```

luckyNumber =

Object

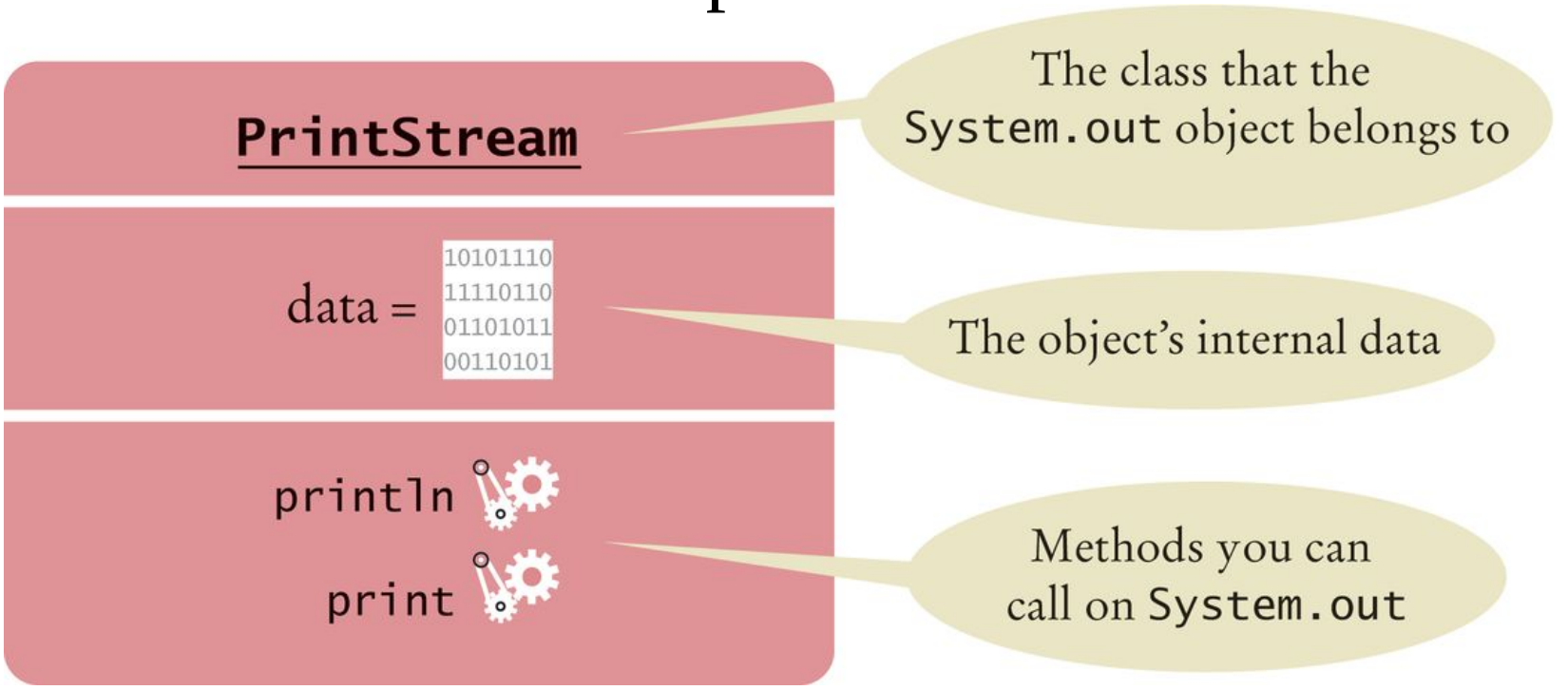
An object is an entity that you can manipulate by calling one or more of its methods.

macrowave

dishwasher

Each object belogs to a different class of objects

A Sample Class



Think of each method (shown as gears) as piece of machinary that carries out its tasks.

Each object is an instance of a class.

The System.out object is an instance of PrintStream class. (more about this later)

Methods

- Method: Sequence of instructions that accesses/manipulates the data of an object
- You manipulate objects by calling its methods
- Class determines legal methods

```
String greeting = "Hello";
greeting.println() // Error
greeting.length() // OK
```

- Public Interface: Specifies what you can do with the objects of a class

Variables

Variable is a storage location in a computer program.

A variables has name and type.

TypeName VariableName = value

Example:

```
double length =22.5;
String greeting= "hello";
```

var keyword (Java 10)

As of Java10, you need not specify the type of the varibable that you intialize, for example

```
double length =22.5;
String greeting= "hello";
```

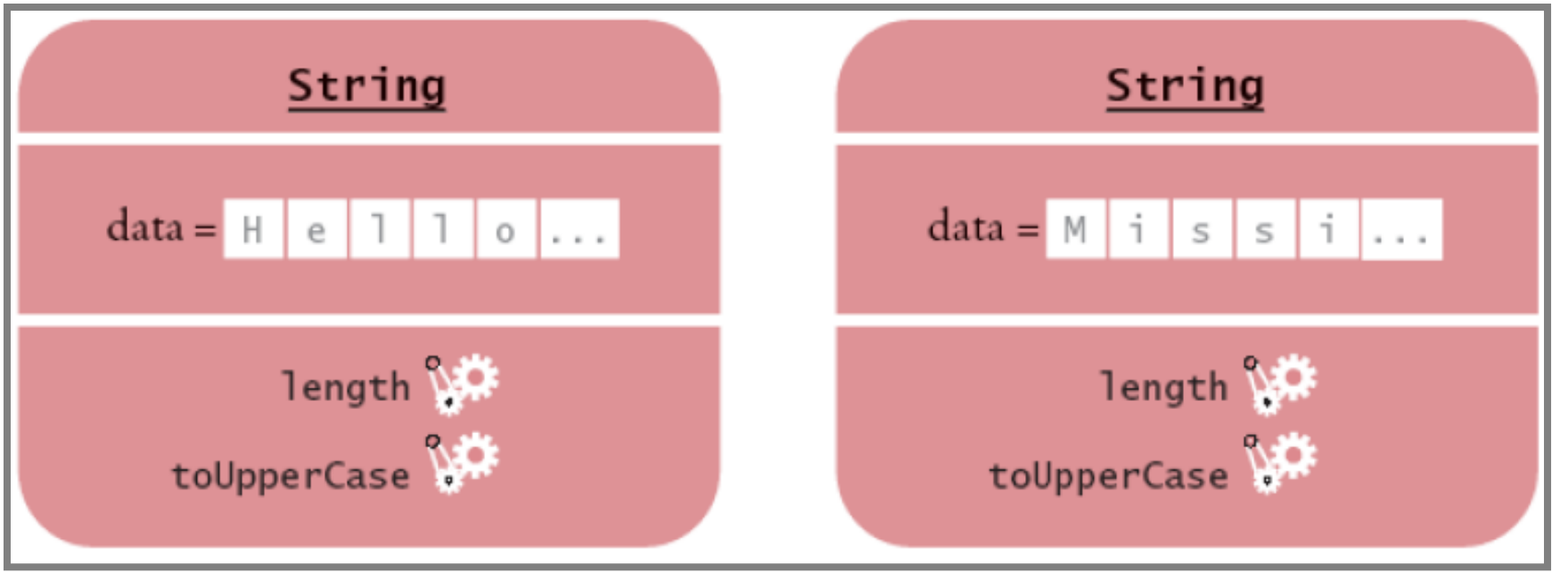
You can write

```
var length =22.5;
var greeting= "hello";
```

The java compiler inferes the type of the variable from the intitial value.

However, **avoid using** this feature of Java.

A Representation of Two String Objects



length(): counts the number of characters in a string.
toUpperCase(): creates another String object that contains the characters of the original string, with all lowercase letters converted to uppercase.

Examples

```
String greeting = "Hello, World!";
int n = greeting.length(); // sets n to 13
```

```
String river = "Mississippi";
String bigRiver = river.toUpperCase();
    // sets bigRiver to "MISSISSIPPI"
```

greeting, river, and bigRiver are instance of class String.

When applying a method to an object, make sure method is defined in the class that object belongs to.

```
String.println(); // This method call is an error
```

Implicit and Explicit Parameters

- Parameter (explicit parameter): Input to a method. Not all methods have explicit parameters.

```
System.out.println(greeting)
greeting.length() // has no explicit parameter
```

- Implicit parameter: The object on which a method is invoked

```
System.out.println(greeting)
```


Continued…

A More Complex Call

`replace(...)` method carries out a search-and-replace operation

```
String river = "Mississippi";
String str = river.replace("issipp", "our")

// equivalent to the previous one
String s = "Mississippi".replace("issipp", "our");
```

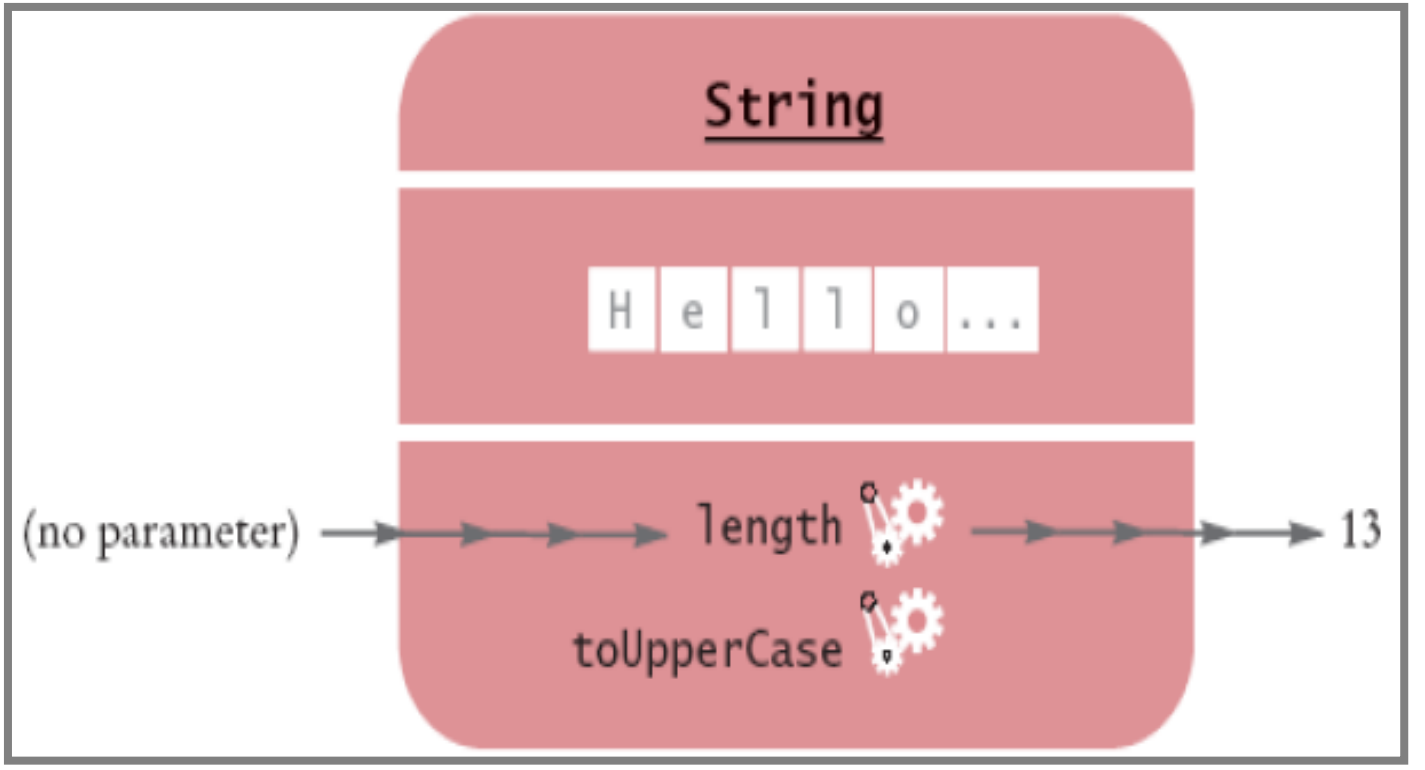
- Two explicit parameters: the strings **`"issipp"`** and **`"our"`**
- One implicit parameter: the string **`"Mississippi"`**
- A return value: the string **`"Missouri"`**

Continued...

Return Values

- Return value: A result that the method has computed for use by the code that called it

```
int n = greeting.length(); // return value stored in n
```



Method Definitions

- Method definition specifies types of explicit parameters and return value
- Type of implicit parameter = current class; not mentioned in method definition
- Example:

No explicit parameter

Explicit parameters

```
public int length( )
   // return type: int
   // no explicit parameter
public String replace(String target, String replacement)
   // return type: String;
   // two explicit parameters of type String
...
```

Method Definitions

- If method returns no value, the return type is declared as `void`

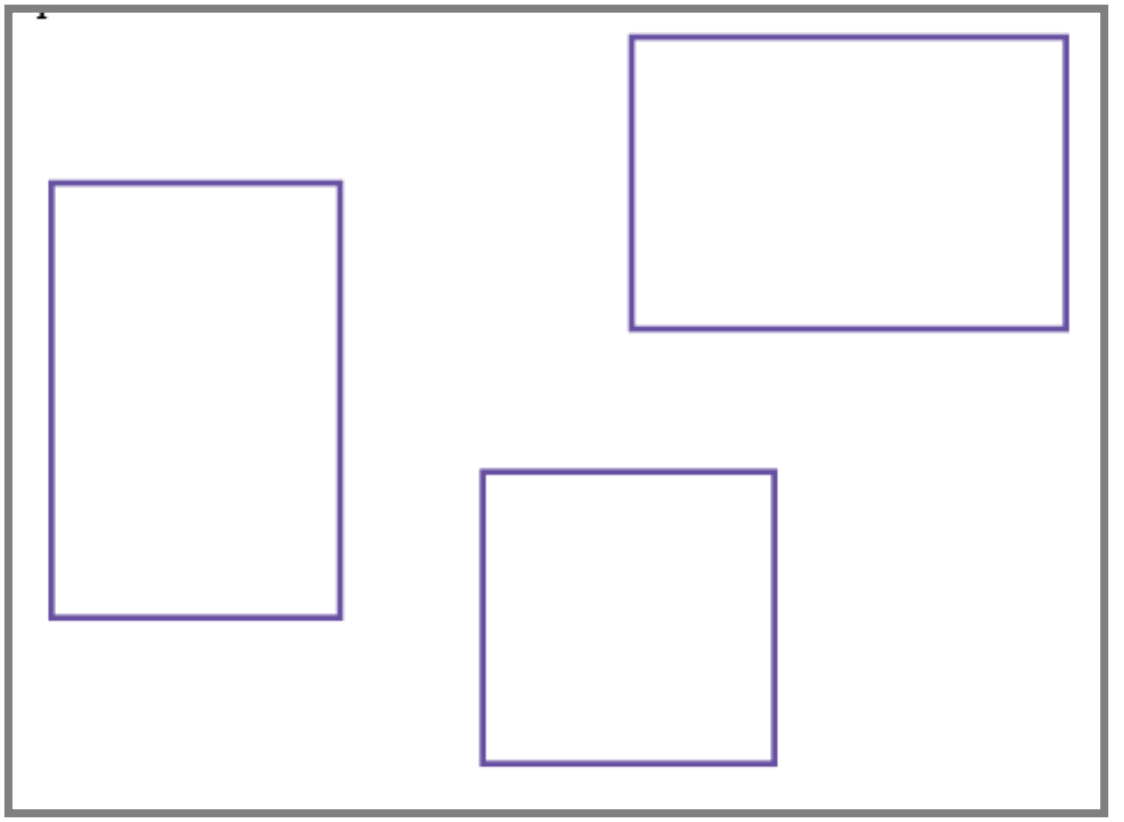
```
public void println(String output) // in class PrintStream
```

- A method name is overloaded if a class has more than one method with the same name (but different parameter types)

```
public void println(String output){ }
public void println(int output){ }
```

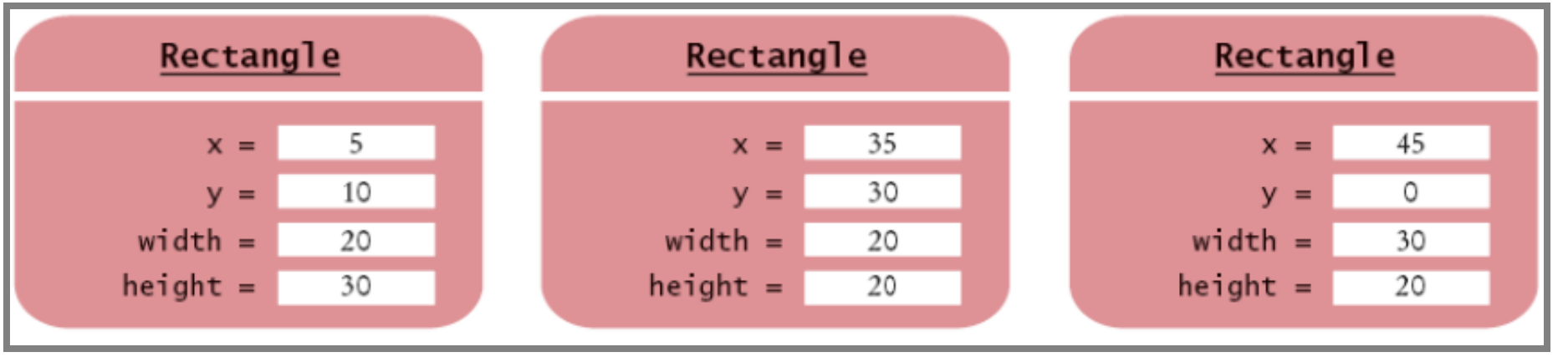
Rectangular Shapes and Rectangle Objects

- Objects of type `Rectangle` *describe* rectangular shapes



Rectangular Shapes and Rectangle Objects

- A `Rectangle` object isn't a rectangular shape– it is an object that contains a set of numbers that describe the rectangle



Constructing Objects

```
new Rectangle(5, 10, 20, 30)
```

Detail:

- The new operator makes a `Rectangle` object
- It uses the parameters (in this case, `5`, `10`, `20`, and `30`) to initialize the data of the object
- It returns the object reference

Usually the output of the new operator is stored in a variable

```
Rectangle box = new Rectangle(5, 10, 20, 30);
```

Constructing Objects

- The process of creating a new object is called *construction*
- The four values `5`, `10`, `20`, and `30` are called the *construction parameters*
- Some classes let you construct objects in multiple ways

```
new Rectangle()
    // constructs a rectangle with its top-left corner
    // at the origin (0, 0), width 0, and height 0
```

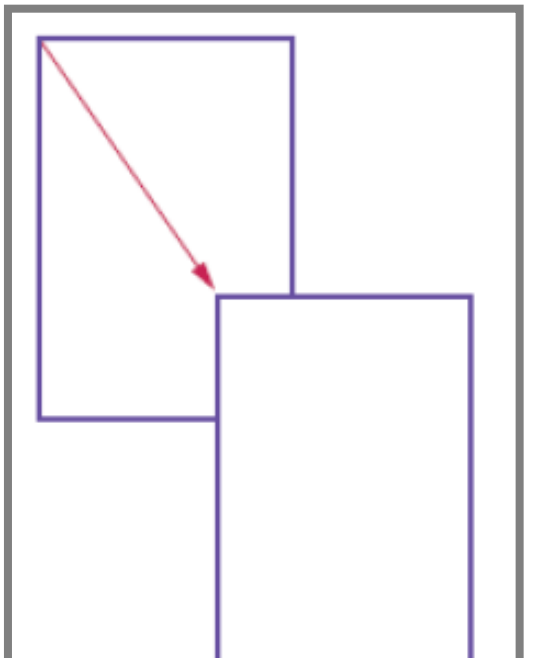
Accessor and Mutator Methods

- **Accessor** method: does not change the state of its implicit parameter

```
double width = box.getWidth();
```

- **Mutator** method: changes the state of its implicit parameter

```
box.setX(15);
box.setY(25);
```



Importing Packages

Don't forget to include appropriate packages:

- Java classes are grouped into packages
- Import library classes by specifying the package and class name:

```
import javafx.scene.shape.Rectangle;
```

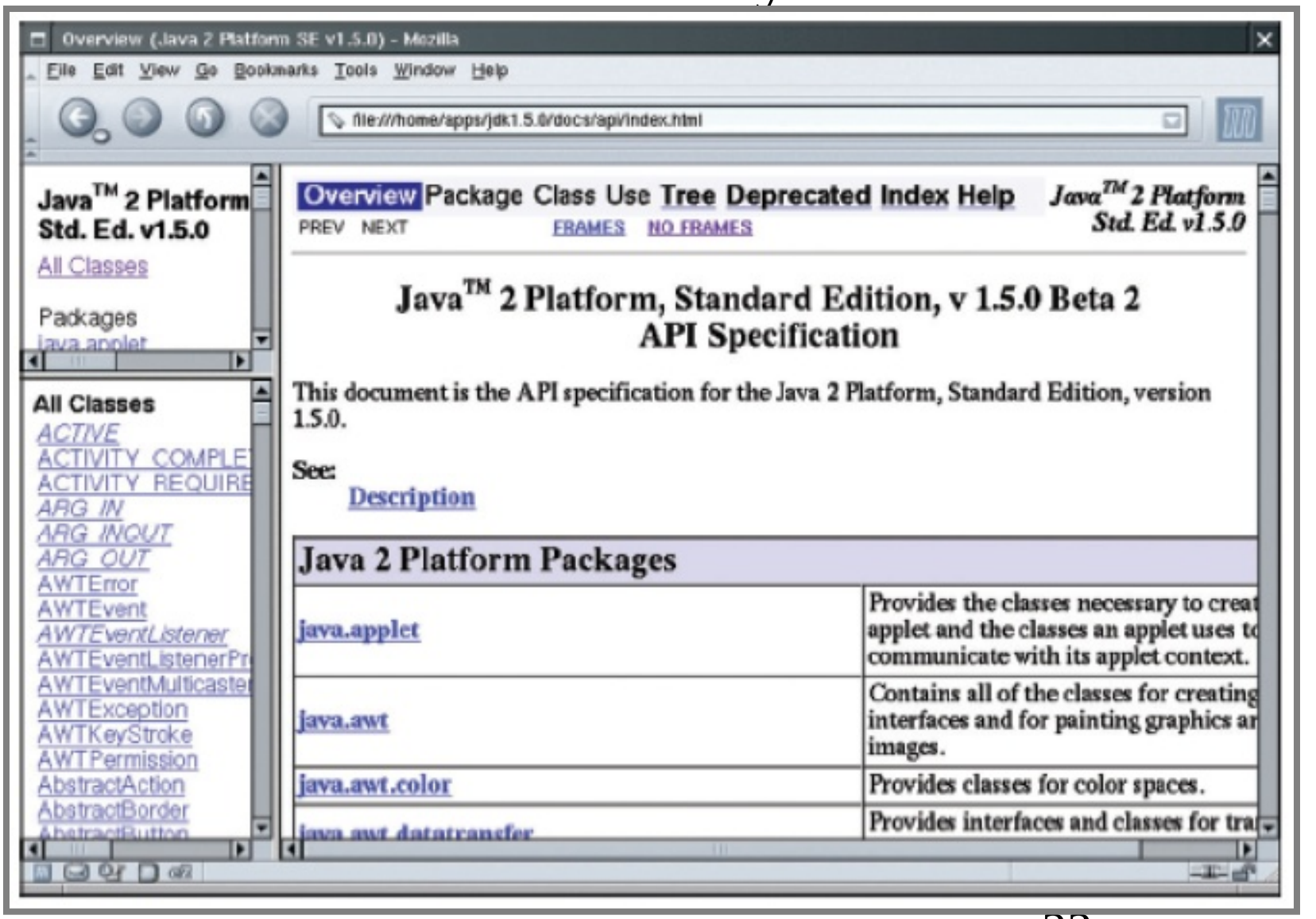
- You don't need to import classes in the `java.lang` package such as `String` and `System`

The API Documentation

- API: Application Programming Interface
- Lists classes and methods in the Java library

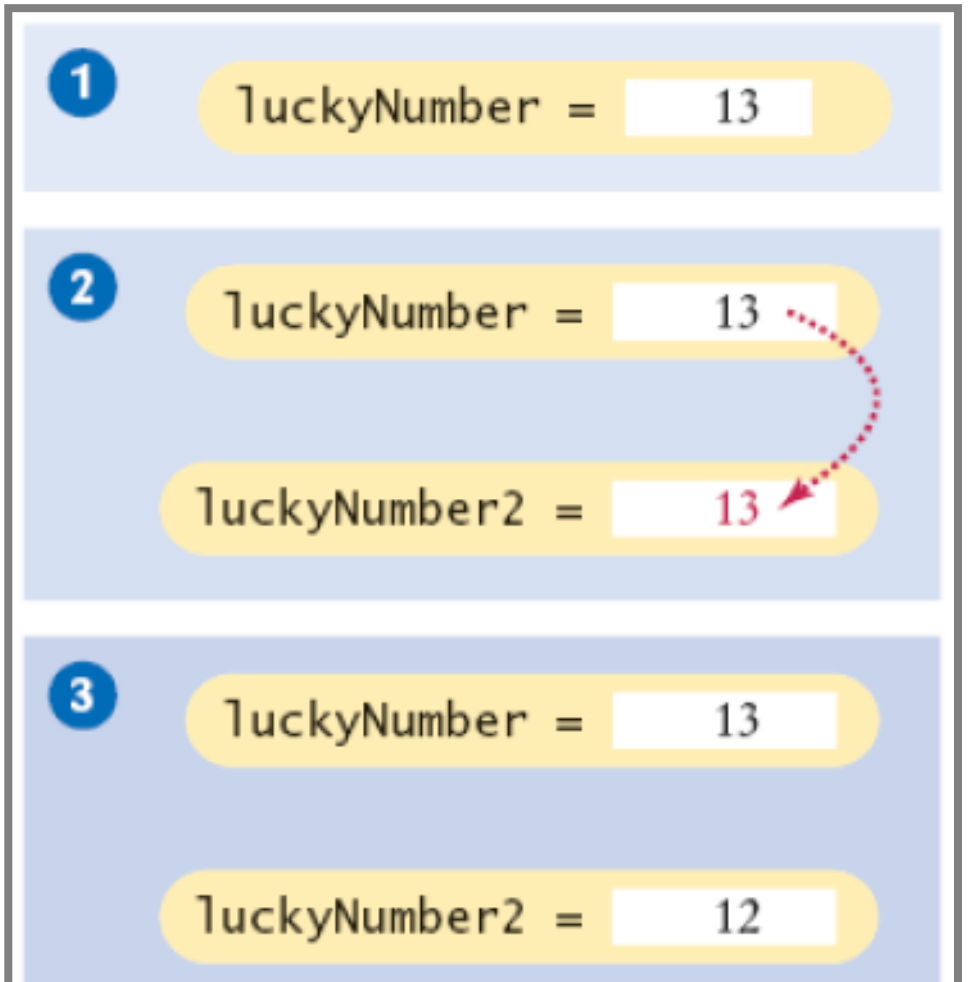
http://docs.oracle.com/javase/8/docs/api/

The API Documentation of the Standard Java Library

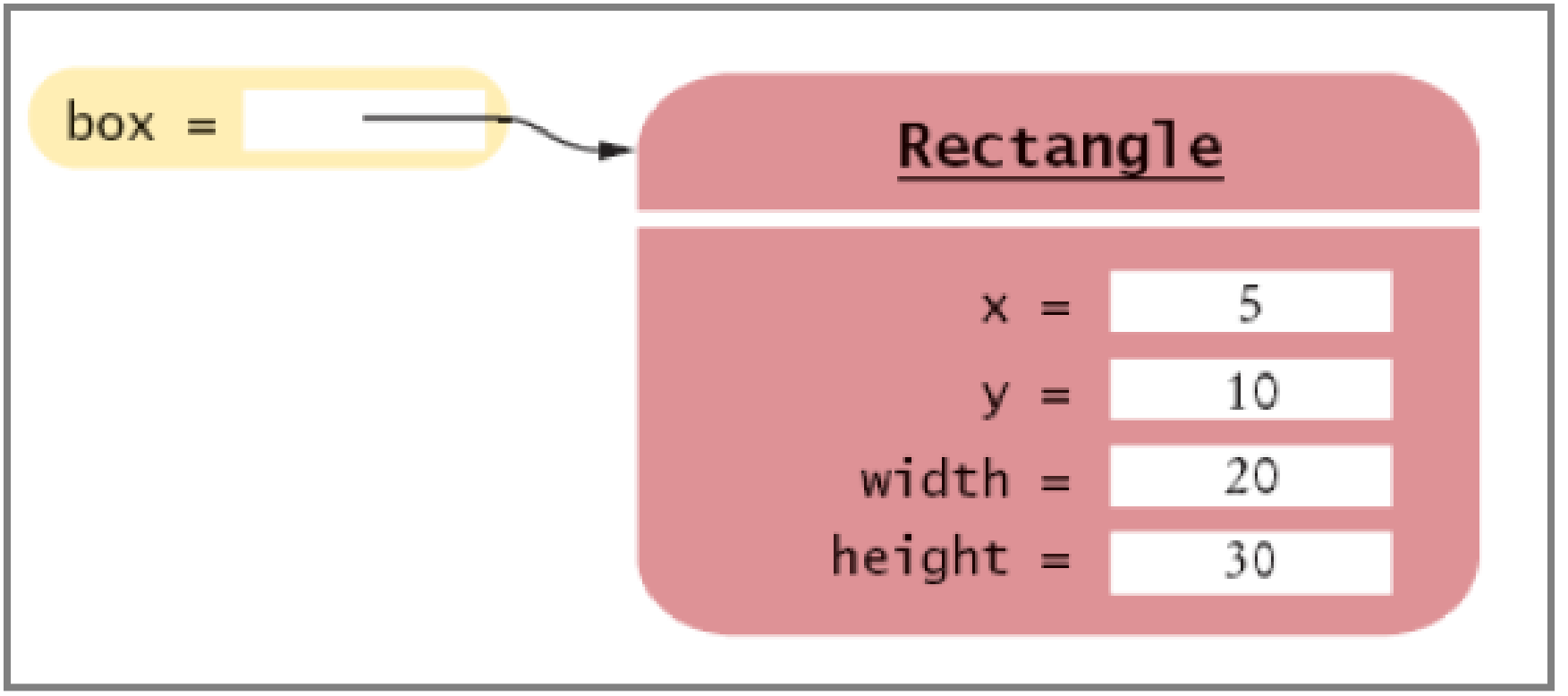


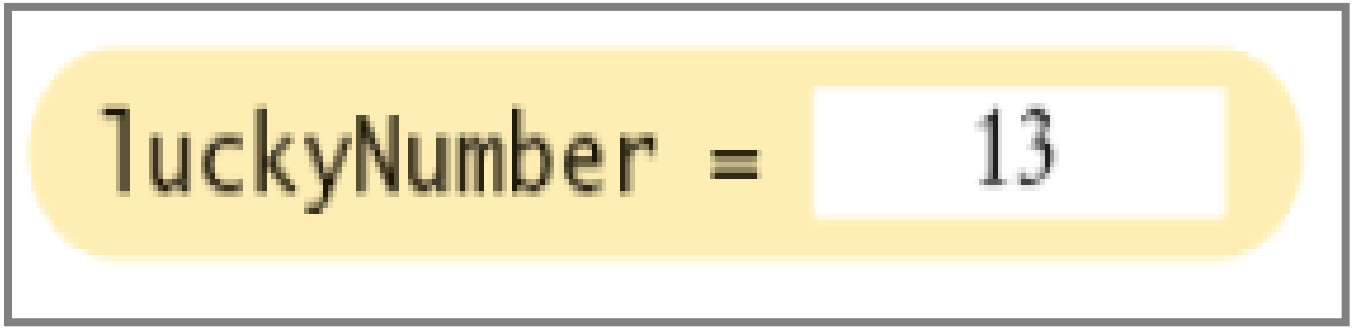
Copying Numbers

```
int luckyNumber = 13;
int luckyNumber2 = luckyNumber;
luckyNumber2 = 12;
```



Object Variables and Number Variables





Object References

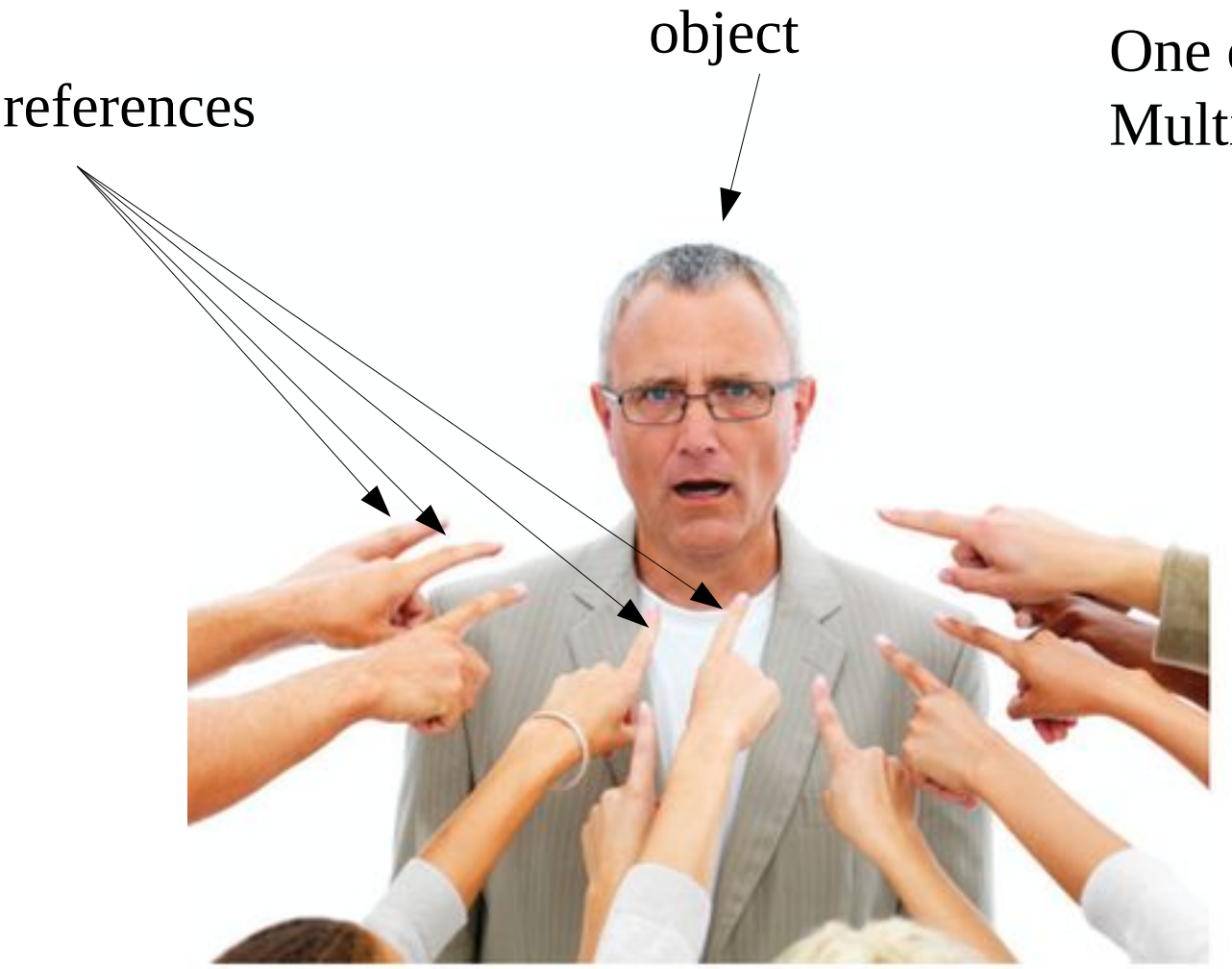
- Reference refers to the location of an objects.
- The **new** operator returns a reference to a new object

```
Rectangle box = new Rectangle();
```

- Multiple object variables can refer to the same object

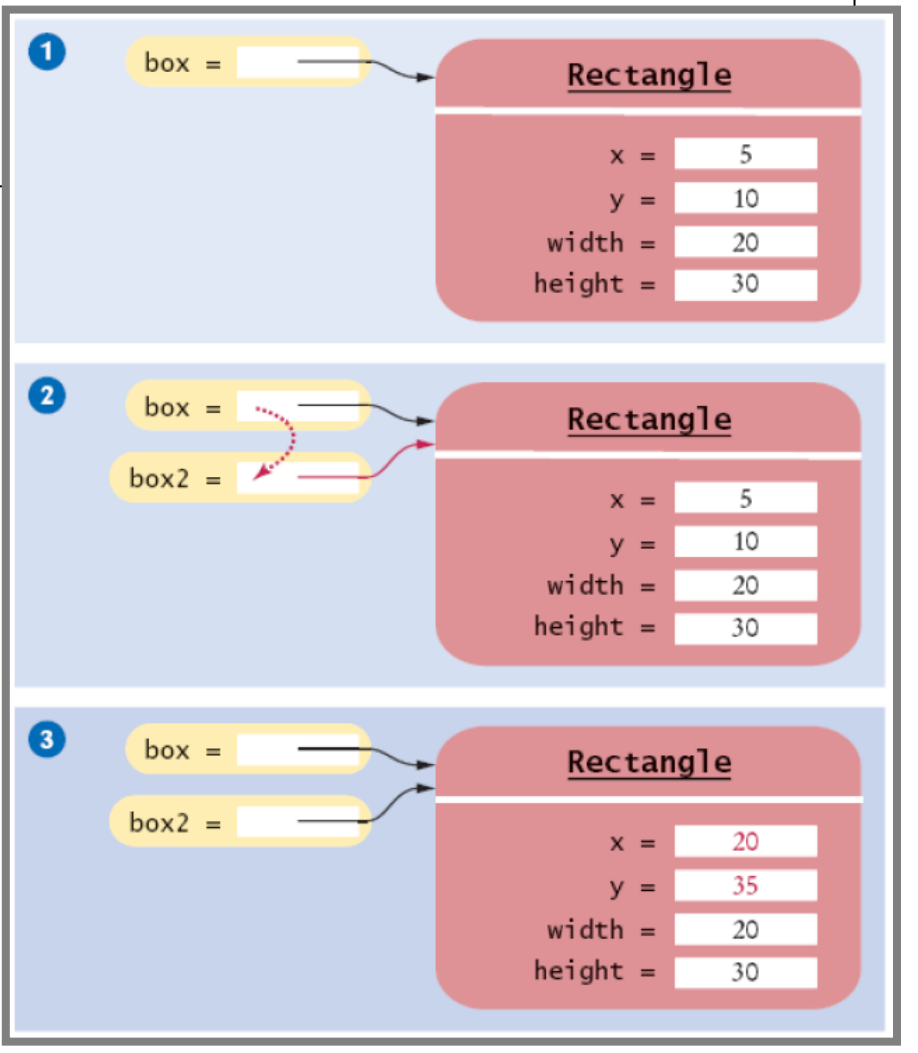
```
Rectangle box = new Rectangle(5, 10, 20, 30);
Rectangle box2 = box;
```

One object
Multiple references



Copying Object References

```
Rectangle box = new Rectangle(5, 10, 20, 30);
Rectangle box2 = box;
box2.setX(20);
Box2.setY(35)
```



Summary

Objects are entities in your program that you manipulate by calling methods.

A method is a sequence of instructions that access/modifies the data of an object.

A class describes a set of objects with the same behavior.

Suggested Exercises

E2.3, E2.4, E2.5, E2.6, E2.7, and E2.8