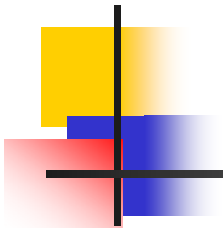
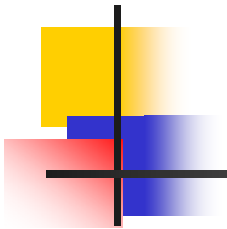
Methods



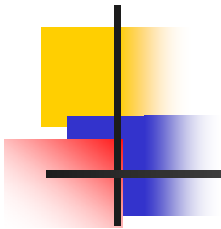
Outline

- What is a method
- Method Structure
- Method invocation
- Method Types
- Local Variables
- Return Statement
- Method Parameters



- What is a method ...

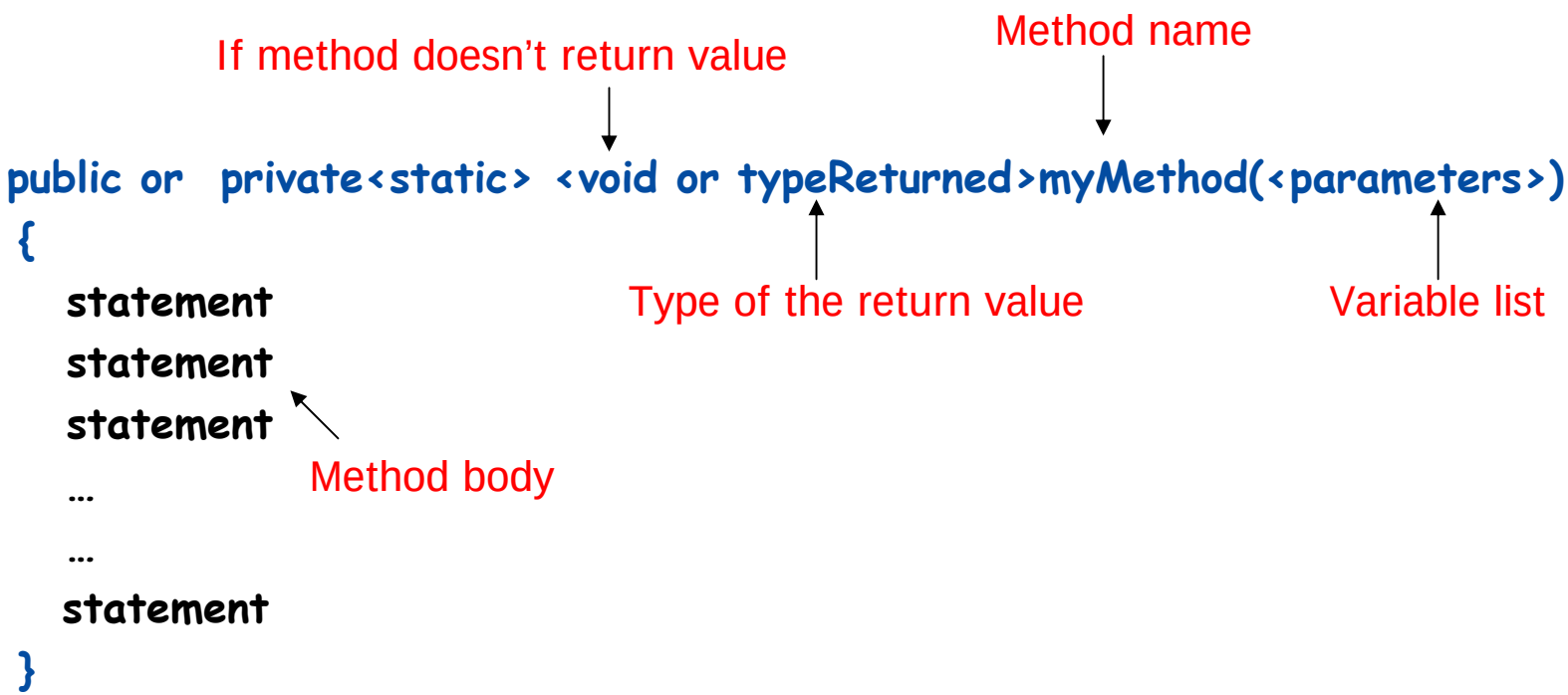
- A Java application program consists of one or more classes.
- Each class has one or more methods
- Each method consists of one or more statements
- Each method has a name.
- One of the methods must be called **main**
  - When a Java application program is run, the *run-time system* automatically invokes the method named **`main`**
  - All Java application programs start with the **`main`** method

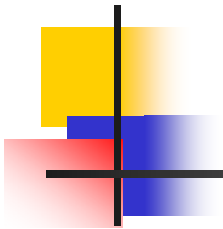


... - What is a method ...

```
public class class-name {
        method1
        method 2
        method 3
        ...
        ...
        method n
}
```

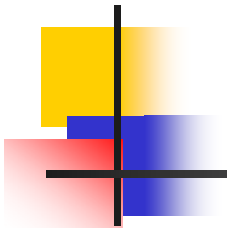
-- Method Structure





- Invoking a Methods

- The statements inside a method body are executed when the corresponding method is called from another method.
- Calling a method is also called invoking a method
- Each time the method is invoked, its corresponding body is executed

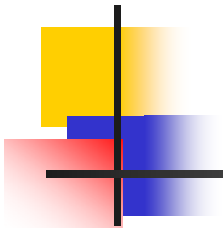


- `return` Statements ...

- The body of a method that returns a value must also contain one or more **return** statements
  - A **return** statement specifies the value returned and ends the method invocation:

```
return Expression;
```

  - **Expression** can be any expression that evaluates to something of the type returned listed in the method heading

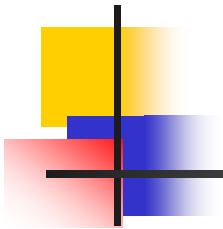


… - `return` Statements

- A **void** method need not contain a **return** statement, unless there is a situation that requires the method to end before all its code is executed

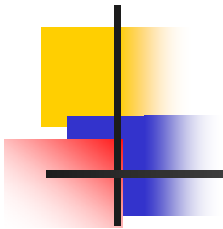
- In this context, since it does not return a value, a **return** statement is used without an expression:

```
return;
```



- Local Variables

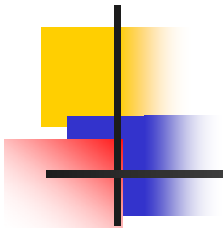
- A variable declared within a method definition is called a *local variable*
  - All variables declared in the **main** method are local variables
  - All method parameters are local variables
- If two methods each have a local variable of the same name, they are still two entirely different variables



- Method Parameters ...

- Methods exchange data using a list of parameters
  - These *parameters* are also called *formal parameters*
- A parameter list provides a description of the data required by a method
  - It indicates the number and types of data pieces needed, the order in which they must be given, and the local name for these pieces as used in the method

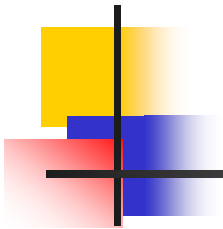
```
public double myMethod(int p1, int p2, double p3)
```



... - Method Parameters ...

- When a method is invoked, the appropriate values must be passed to the method in the form of *arguments*
  - Arguments are also called *actual parameters*
- The number and order of the arguments must exactly match that of the parameter list
- The type of each argument must be compatible with the type of the corresponding parameter

```
int a=1,b=2,c=3;
double result = myMethod(a,b,c);
```

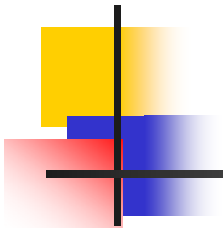


... - Method Parameters ...

- In the preceding example, the value of each argument (not the variable name) is plugged into the corresponding method parameter
  - This method of plugging in arguments for formal parameters is known as the *call-by-value mechanism*

... - Method Parameters ...

- If argument and parameter types do not match exactly, Java will attempt to make an automatic type conversion
  - In the preceding example, the **`int`** value of argument **`c`** would be cast to a **`double`**
  - A primitive argument can be automatically type cast from any of the following types, to any of the types that appear to its right:



... - Method Parameters

- A parameters is often thought of as a blank or placeholder that is filled in by the value of its corresponding argument
- However, a parameter is more than that: it is actually a local variable
- When a method is invoked, the value of its argument is computed, and the corresponding parameter (i.e., local variable) is initialized to this value
- Even if the value of a formal parameter is changed within a method (i.e., it is used as a local variable) the value of the argument cannot be changed

THE END