Foundation of Programming

User Define Method

CAMT
Digital School

Writing Your Own Methods

- Like Java's methods, a method that you create:
- Has a *name*
- May accept *arguments*
- May return a *value*
- May be used as part of an expression.

CAMT Digital School

When to use methods ?

- The similar code comes again and again
  - With the same operation by may be different value
- Code reusable

CAMT Digital School

```java
double area = 0;
area = Math.PI * Math.pow(5, 2);
System.out.println("The area is " + area);
area = Math.PI * Math.pow(7, 2);
System.out.println("The area is " + area);
```

```java
double area = 0;
area = getCircleArea(5);
System.out.println("The area of circle is " + area);
area = getCircleArea(7);
System.out.println("The area of circle is " + area);
```

```java
public static double getCircleArea(double radias){
return Math.PI * Math.pow(radias, 2);
}
```


CAMT Digital School

Methods that Return a Value

- The number of calories used while running depends on the runner's weight as well as the distance that he/she has run
- A common rule of thumb used to estimate the number of calories burned is:

```
calories = .653 × weight × distance
```

where `weight` is the runner's weight in pounds

`distance` is in miles.

CAMT Digital School

Methods that Return a Value

- Write a program that calculates the number of calories burned as a function of weight and distance.
- Include a method:

```
double caloriesBurned(double weight, double distance)
```

accepts two arguments of type double, and returns a value of type double.

The method *double* caloriesBurned(double weight, double distance)

Methods that Return a Value

```
import java.util.*;
public class RunnersCalculator
{

    public static double caloriesBurned(double weight, double distance)
    {

      // returns the number of calories burned using the formula
      // calories = .653 × weight × distance

        double calories = .653* weight *distance;
        return calories;
    }
}
```


CAMT Digital School

Methods that Return a Value

```
    public static void main(String[] args)
     {
         Scanner input;
         double myWeight, myDistance, totalCalories;

         input = new Scanner(System.in);
         System.out.print("Enter weight in pounds: ");
         myWeight = input.nextDouble();

         System.out.print("Enter distance in miles: ");
         myDistance = input.nextDouble();

         totalCalories =  caloriesBurned(myWeight, myDistance);
         System.out.println("Calories burned: "+ totalCalories);
     }
}
```

Output

Enter weight in pounds: 165.0

Enter distance in miles: 6.0

Calories burned: 646.47

CAMT Digital School

Discussion

- Like all Java applications, `RunnersCalculator`
  - begins execution with `main(...)` (lines 11 - 22)
  - Within `main(...)` there is a call to the method `caloriesBurned(...)` on line 24:

```
24. totalCalories =  caloriesBurned(myWeight, myDistance);
```

- `caloriesBurned(...)` has two arguments:
  - myWeight and
  - myDistance;
- the returned value is assigned to the variable *`totalCalories`*.

CAMT Digital School

- The instructions of the method `caloriesBurned(...)` are specified on lines 9 and 10:

```
double calories = .653* weight *distance;
return calories;
```

- Line 5 is the header of the method:

```
public static double caloriesBurned(double weight, double distance)
```

- For now, ignore the keywords *`public`* and *`static`*. The remainder of the header specifies:
  - The data type of the return value: `double`
  - The name of the method: `caloriesBurned`
  - The parameters: `weight` and `distance`

CAMT Digital School

Methods that Return a Value

- The parameters specify the *type* and *number* of the arguments that must be *passed* to the method
- When this method is invoked with two arguments
  - the value of the first argument is assigned or passed to weight
  - the value of the second argument is passed to parameter distance.
- If the method call is

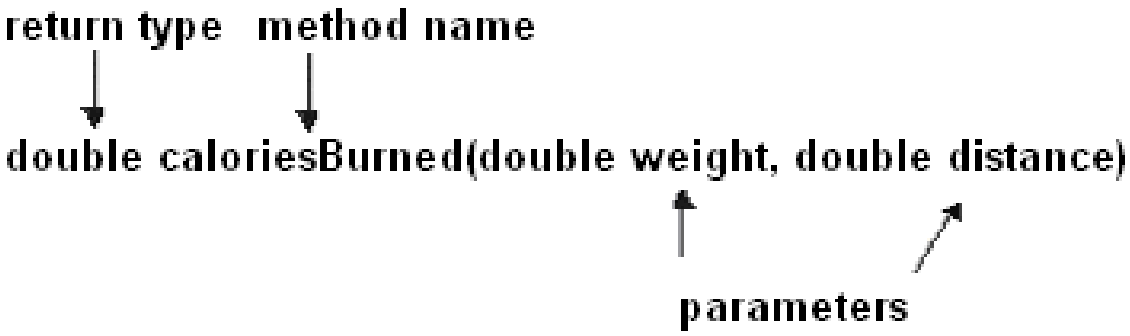
```
caloriesBurned(155.5, 3.5)
```

the parameter `weight` gets the value 155.5

the `distance` gets the value 3.5.

CAMT Digital School

Methods that Return a Value



Parts of a *method header*

The block consisting of lines 6 through 11 contains the instructions of the method `caloriesBurned(...)`.

Line 9 is an expression that calculates the number of calories burned.

Method that Return a Value

Line 10 is a *return* statement. The return statement has the form:

```
return expression
```


CAMT Digital School

The return statement

- The return statement has two purposes:
  - Specify the *value that the method returns* to the caller.
  - *Terminate the method* and returns program control to the caller

CAMT Digital School

A trace of RunnersCalculator

myWeight myDistance totalCalories	Line 15: double myWeight, myDistance, totalCalories
165.0 myWeight myDistance totalCalories	Line 19: myWeight = input.nextDouble();.
165.0 6.0 myWeight myDistance totalCalories	Line 22: myDistance = input.nextDouble();
165.0 6.0 myWeight myDistance totalCalories	Line 24: totalCalories = **caloriesBurned(myWeight,myDistance);** Call caloriesBurned(…). Pass values of the arguments myWeight and myDistance to parameters weight and distance, respectively.

Program control passes to `caloriesBurned(...)`

A trace of RunnersCalculator

165.0 weight 6.0 distance	Line 5: **public static double caloriesBurned(double weight, double distance)** The parameters weight and distance are initialized with the values of arguments myWeight and myDistance.
165.0 weight 6.0 distance 646.47 calories	Line 9: **double calories = .653* weight *distance;** Declare the variable calories. Calculate the number of calories burned, and initialize calories to that value.
165.0 weight 6.0 distance 646.47 calories	Line 10: **return calories;** Return the value of calories to the caller and exit.

Program control returns to the assignment on line 20

165.0 myWeight 6.0 myDistance 646.47 totalCalories	Line 24 (resumed): Assign the returned value to totalCalories.
165.0 myWeight 6.0 myDistance 646.47 totalCalories	Line 21: Print the results.

CAMT Digital School

Void Methods

- A method can perform a task without returning a value.
- Such a method is called a *void* method:

```
void drawSquare( int size)
```

- might be the header of a method that draws a square on the screen and *does not return a value*

CAMT Digital School

Void Methods

- The expression:

```
5* Math.sqrt(25)
```

- is certainly meaningful and has the value 25.0, but:

```
5*drawSquare(25)
```

- makes no sense since `drawSquare(25)` does not return a value.

CAMT Digital School

Void Methods

- A call to a void method is a “stand-alone” statement consisting of
  - the method name
  - arguments that must be passed to the method

```
System.out.println(“Print me!”);
```

or

```
drawSquare(10);
```


CAMT Digital School

Void Methods

Problem statement:

Write a program that includes a void method:

```
void coinChanger(int amount)
```

that accepts a single integer argument between 1 and 100 that

represents an amount of money between $.01 and $1.00

The method makes change for that amount using the minimum number of coins.

Coins are in denominations of half dollars, quarters, dimes, nickels, and pennies.

CAMT Digital School

Solution

```
import java.util.*;
public class MoneyChanger
{
    public static void coinChanger (int amount)
    {
        // calculates the minimum number of half dollars,
       quarters, dimes,
        // nickels and pennies in amount
        int halfDollars, quarters, dimes, nickels, pennies;

        System.out.println();
        System.out.println(amount+" cents can be converted to:");

        halfDollars = amount/50; //determine number of half
                                 dollars
```

Solution

```
amount = amount%50;     // how much remains?
quarters = amount/25;   // determine number of quarters
amount = amount%25;     // how much remains?
dimes = amount/10;      // determine the number of dimes
amount= amount%10;      //  how much remains?
nickels = amount/5;     // determine the number of nickels
pennies = amount%5;     //remainder is the number of
                        pennies
```


CAMT Digital School

Solution

```
        System.out.println("Half Dollars: "+ halfDollars);
        System.out.println("Quarters: "+ quarters);
        System.out.println("Dimes: "+ dimes);
        System.out.println("Nickels: "+ nickels);
        System.out.println("Pennies: "+ pennies);
        return;        // return statement is optional here
}
```


CAMT Digital School

Solution

```
public static void main(String[] args)
 {
        Scanner input;
        input = new Scanner(System.in);
        System.out.print("Enter a value between 1 and 100: ");
        int money = input.nextInt();
        coinChanger(money);     //call to method coinChanger
  }
}
```

Output

Enter a value between 1 and 100: 83

83 cents can be converted to:

Half Dollars: 1

Quarters: 1

Dimes: 0

Nickels: 1

Pennies: 3

CAMT Digital School

Void Methods

- Because `coinChanger(…)` does not return a value
- The call to `coinChanger(...)` is not called *within* an expression.
- The method call is the Java statement (line 32):

```
coinChanger (money);
```

- The return statement on line 24
  - does not include a return value or an expression.
  - no value is returned to the calling method.

CAMT Digital School

Void Methods

- The *return* statement on line 24 is unnecessary.
- After a void method executes its last statement, the method automatically returns
  - no final return statement is necessary.

CAMT Digital School

Putting It All Together

- A Java method consists of a:
  - header followed by a
  - method block.
- The parameters in the header specify the number and type of the arguments that must be passed to the method.
- When a method is invoked, the values stored in the arguments are *copied* to the parameters

CAMT Digital School

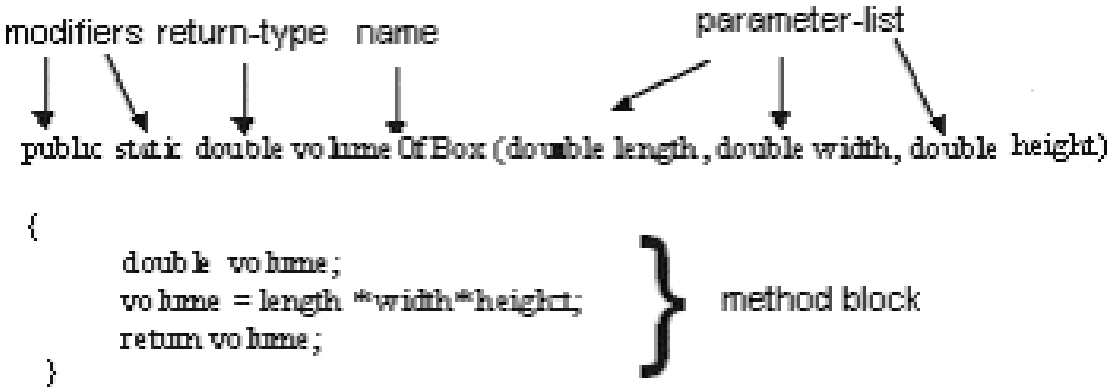
Putting It All Together

- The method block is a sequence of *statements enclosed* by curly braces:

```
{
        statement-1;
        statement-2;
        statement-3;
                …
        statement-n;
}
```


CAMT Digital School

A method that calculates the volume of a box



Putting It All Together

<u>Method Name</u>

- The name of a method must be a valid Java identifier
- Standard Java convention specifies that the name of a method
  - *begins* with a lowercase letter
  - starts each succeeding word in the method name with an uppercase letter
- For example
  - the names `volumeOfBox` and `caloriesBurned` *both follow this convention*
  - the names `VolumeOfBox` and `volumeofbox` *do not*.

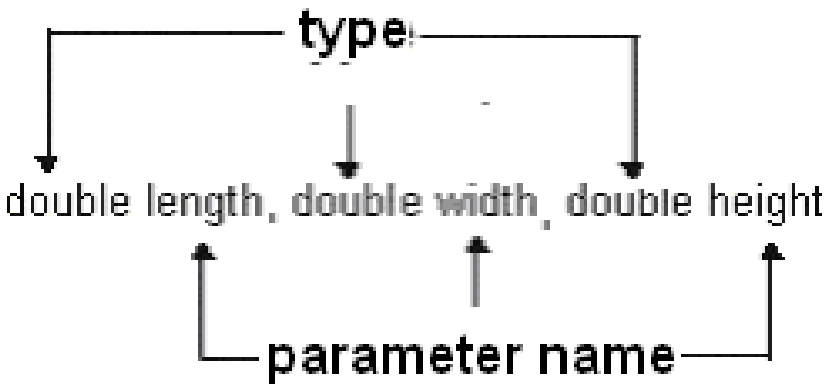
CAMT Digital School

Putting It All Together

<u>Parameter-List.</u>

A method's parameter-list consists of *pairs* of the form:

```
type   parameter-name
```



Putting It All Together

<u>Argument Passing.</u>

- When calling a method
  - the caller passes arguments to the *parameters*
- The calling statement must provide a *type-suitable value* for each parameter
  - If a method has five parameters, *five arguments* are required.

CAMT Digital School

Putting It All Together

<u>Pass by Value.</u>

- All arguments are passed "*by value*."
- This means that the arguments are evaluated and values of the arguments are *copied* to the parameters of a method
  - Modifying the parameters in the method has no effect on the value of any variables passed as arguments.

CAMT Digital School

Putting It All Together

Method Block.

- The statements of the method-block accomplish the task of the method.

CAMT Digital School

Putting It All Together

<u>The return Statement.</u>

- A method that returns a value must include a *return* statement. The form of the return statement is

```
return expression
```

- When a method executes the return statement,
  - the method terminates,
  - program control passes back to the *caller*, and
  - any statements *following the return statement* are ignored

CAMT Digital School

Putting It All Together

<u>Local Variables.</u>

- Variables that are declared within a method are called the local variables of that method
- Local variables *exist* and are known only within the *method* in which they are declared
- When a method *exits*, the local variables are *destroyed*
- Local variables do not exist beyond the life of a method call

CAMT Digital School

Local Variables

- When a method is invoked
  - Memory *for local variables* is allocated
  - When a method *exits*, that memory is *de-allocated*.
- A method's local variables do not retain values from call to call
  - When a method exits, its local variables no longer exist
- The concept of local variables is tied to the broader topic of scope
  - which we discuss in the next section.

CAMT Digital School

Scope

- The scope of a variable is that section of the program in which a variable can be *accessed* or *referenced*.
- For example, consider the following void method that computes the `sum` and product of the first `n` positive integers:

CAMT Digital School

Scope

```
void sumAndProduct(int n)
{
    int sum = 0;
    int product = 1;
    for (int  i= 1; i<= n; i++)
    {
        sum += i;
        product *= i;
    }
        System.out.println( "Sum of the  first " + n+ "
       positive integers is "+ sum);
        System.out.println("Product of the  first " + n+ "
              positive integers is "+ product);
 }
```

Scope

- The method `sumAndProduct` has several local variables: `n`, `sum`, `product`, and `i`. The scope of each of these variables is as follows:
  - The scope of parameter `n` is the entire method.
  - The scope of `sum` begins with its declaration on line 3 and extends to the end of the method.
  - Similarly, the scope of product extends from its declaration on line 4 to the method's end.
  - The variable `i` does not exist beyond the block of the for-loop.
  - Thus, the scope of variable `i` is lines 5 through 9. Outside of the for-loop, `i` is inaccessible and unknown.

CAMT Digital School

Scope

- In general, the scope of a *variable* begins with its declaration and extends to the end of the block in which it is declared.

CAMT Digital School

Scope

- A block is a group of statements
  - enclosed by curly braces *{ and };*
- if you declare a variable in the *outermost* block of a method
  - Its scope extends from the declaration to the end of the method
  - Otherwise, the scope of a variable declared within an inner or nested block
    - Begins at the declaration
    - Terminates at the end of that block.

CAMT Digital School

Scope

```
if (purchase > 200)
{
        double discount = .20* purchase;
        double discountPrice = purchase-discount;
        tax = .05*discountPrice;
        total = discountPrice + tax;
}
else
{
        tax  = .05*purchase;
        total = purchase + tax;
}
```

- The scope of the variables `discount` and `discountPrice` extends from their definitions to the end of the "if block." Thus, neither variable is known within the "else block."

CAMT Digital School

Scope

- The scope of a variable declared in the header of a for loop is the entire for loop. In the segment

```
for (int i = 0; i<= 50; i++)
{
        //statements
}
```

- The control variable `i` is unknown once the loop terminates.

CAMT Digital School

Multiple return statements

- A method may have more than one return statement
  - only one executes before the method terminates.

- The first return statement that executes terminates the method

CAMT Digital School

Multiple return statements

- Example

- A prime number p is a positive integer greater than 1 that has no positive integer divisors other than 1 and p
- For example 101 is a prime number since no positive integers other than 1 and 101 divide 101 evenly
- The integers 2, 3, 5, 7, and 37 are all prime numbers
- On the other hand, 100 is not a prime number because 5 is a divisor of 100. With the exception of 2, all prime numbers are odd.

CAMT Digital School

Multiple return statements

- Problem statement:
- Write a program that prompts a user for a positive integer and determines whether or not the number is prime. Include a method

```
boolean  isPrime(int p)
```

that accepts an integer `p` as a parameter

returns true if `p` is prime; otherwise false.

The isPrime method

Multiple return statements

```
import java.util.*;
public class PrimeChecker
{
    public static boolean isPrime(int p)
                //returns true if p is a prime number
    {
      1.  if (p <=1) // 0, 1, and all negatives are not prime
      2.      return false;
      3.  else if (p == 2) // if p is 2; return true (exit)
                              because 2 is prime
      4.      return true;
      5.  else if ( p % 2 == 0) // if p is even and not 2 ,
                                      return false (exit);
      6.      return false;
```

Multiple return statements

```
        // so  p  is odd;  check for odd divisors
        // if a divisor is found,  return false and exit
    7.     for (int i = 3; i < p; i+=2)    //  i =  3,5,7,9
    8.          if (p % i == 0)// if p % i == 0 then i divides
                                p so p is not prime
    9.             return false;
        // if the method reaches this point, p is prime,
    10.      return true;
}
```


CAMT Digital School

Multiple return statements

```
public static void main(String[] args)
    {
        int number;
        Scanner input;
        input = new Scanner(System.in);

        System.out.print("What number would you like to test? ");
        number = input.nextInt();
        if (isPrime(number))
            System.out.println(number + " is a prime number");
        else
            System.out.println(number+" is not prime");
    }
}
```

Output

What number would you like to test? 6317
6317 is a prime number

What number would you like to test? 7163
7163 is not prime

CAMT Digital School

Discussion

- The method `isPrime(...)` contains no less than five return statements. When any one return statement executes, the method exits and program control passes back to the caller.
- For example:
  - If parameter p has the value 22, the condition on line 5 is true and the return statement on line 6 executes returning false and terminating the method;
  - If p has the value 35, the loop of line 7 executes and when i attains the value of 5, the return on line 9 executes returning false (because 35 % 5 == 0, i.e., 35 is divisible by 5);
  - If p is 23, then none of the conditions of the else-if statement is true nor does the condition on line 8 evaluate to true. Consequently, the return statement on line 10 returns true, i.e., 23 is prime.

CAMT Digital School

Example

"Write a program to get 2 integers and return the addition of the number"

- What is the input?
- What is the output?
- What is the process?

CAMT Digital School

Example

"Write a program to get 2 integers and return the addition of the number if the inputs are positive. Otherwise, return 0."

- What is the input?
- What is the output?
- What is the process?

CAMT Digital School

Example

"Write a program to get a integer and return 2 power by the number."

CAMT Digital School

Example

"Write a program to get 2 integers and return the first number power by the second number."

CAMT Digital School

Example

"Write a program to get an non-negative integers and return the factorial of the number"

CAMT Digital School

Example

"Write a program to get an non-negative integers. In this program, you must have a method to check whether the input is odd number or even number."

CAMT Digital School

Example

"Write a program to get 5 doubles and return the average value."

- You have to send the array to the user-defined method.

CAMT Digital School

"Create a simple calculator that can add, subtract, multiply and divide the input"

- Each operation must be in method
- You need to create the menu for user to choose.

CAMT Digital School