Computer Science 101 S1

Lecture 7

Contents

More String instance methods
Defining methods: parameters, return values, return statement

Coursebook: §10

Review

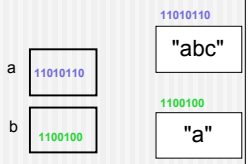
aabc

```
String a, b;

a = new String("abc");
b = a;

a = b.substring(0,1);

System.out.println(a + b);
```



Review

We have seen ***static*** methods:

`<Classname>.<methodName>(<parameters>)`

```
String s = Keyboard.readInput();
int number = Integer.parseInt(s);
int num1 = Math.max(23, 21);
int num2 = Math.max(2, 5);
int num3 = (int)(Math.random() * 3);
```


Review

We have also seen ***instance*** methods:

size1 is 7
word2 "com"
size2 is 3
c1 is 'm'
pos is 1

`<object variable>.<methodName>(<parameters>)`

```
String word1 = "Welcome";
int size1 = word1.length();
String word2 = word1.substring(3, size1-1);
int size2 = word2.length();
char c1 = word2.charAt(2);
int pos = word1.indexOf('e');
```


Other String instance methods

happy HAPPY

toUpperCase()

```
String word1 = "happy";
String word2 = word1.toUpperCase();
System.out.println(word1 + " " + word2);
```

toLowerCase()

```
String word3 = "IMagINE";
String word4 = word3.toLowerCase();
System.out.println(word3 + " " + word4);
```

IMagINE imagine

7

Other String instance methods

trim()

```
String word1 = "  over and out  ";
System.out.println("***" + word1 + "***");
int length1 = word1.length();
word1 = word1.trim();
int length1 = word1.length();
System.out.println("***" + word1 + "***");
System.out.println(length1 + ", " + length2);
```

```
***  over and out  ***
***over and out***
16, 12
```

8

Strings

```
***  SING ***
***SING***
```

Applying a method to a String instance always creates a new String object.

```
String word1 = " sing ";
word1 = word1.toUpperCase();
System.out.println("***" + word1 + "***");
word1 = word1.trim();
System.out.println("***" + word1 + "***");
```

9

Methods

We know how to ***call*** methods that have already been defined for us

```
String word = "Welcome";
int size = word.length();
int pos = word.indexOf("ME");
```

We know how to define ***one*** method called **start()** in our programs

```
public class MyProgram {
     public void start() {
     }
}
```

10

Why methods?

Consider a cook book which contains recipes for making bread:

11

Recipe 1: Rectangular loaf

Put half of the water in a small bowl and add the sugar. Stir until the sugar is dissolved. Sprinkle the yeast over the water and leave for 10 minutes. Combine the flour and salt in a large bowl. Make a well in the flour and add the yeast mixture. Mix thoroughly. Place the dough in a greased bowl, cover and store in a warm place for 30 minutes or until the dough has risen. Knead for another 2 minutes. Divide the mixture in half and put each half into a rectangular baking tin. Wait 10 more minutes for the dough to rise again and bake at 200 degrees for 40 minutes. The bread should be golden brown and should sound hollow if you tap the crust.

12

Recipe 2: Round loaf

Put half of the water in a small bowl and add the sugar. Stir until the sugar is dissolved. Sprinkle the yeast over the water and leave for 10 minutes. Combine the flour and salt in a large bowl. Make a well in the flour and add the yeast mixture. Mix thoroughly. Place the dough in a greased bowl, cover and store in a warm place for 30 minutes or until the dough has risen. Knead for another 2 minutes. Divide the mixture in half. Shape each half into a ball and place on a baking tray. Wait 10 more minutes for the dough to rise again and bake at 200 degrees for 40 minutes. The bread should be golden brown and should sound hollow if you tap the crust.

13

Recipe 3: Long rolls

Put half of the water in a small bowl and add the sugar. Stir until the sugar is dissolved. Sprinkle the yeast over the water and leave for 10 minutes. Combine the flour and salt in a large bowl. Make a well in the flour and add the yeast mixture. Mix thoroughly. Place the dough in a greased bowl, cover and store in a warm place for 30 minutes or until the dough has risen. Knead for another 2 minutes. Divide the mixture into a dozen equally sized pieces. Shape each piece like a sausage-roll and place them spaced out on a baking tray. Wait 10 more minutes for the dough to rise again and bake at 200 degrees for 40 minutes. The bread should be golden brown and should sound hollow if you tap the crust.

14

Recipe 4: Round rolls

Put half of the water in a small bowl and add the sugar. Stir until the sugar is dissolved. Sprinkle the yeast over the water and leave for 10 minutes. Combine the flour and salt in a large bowl. Make a well in the flour and add the yeast mixture. Mix thoroughly. Place the dough in a greased bowl, cover and store in a warm place for 30 minutes or until the dough has risen. Knead for another 2 minutes. Divide the mixture into a dozen equally sized pieces. Shape each piece into a ball and place them spaced out on a baking tray. Wait 10 more minutes for the dough to rise again and bake at 200 degrees for 40 minutes. The bread should be golden brown and should sound hollow if you tap the crust.

15

Recipe 5: Mouse loaf

Put half of the water in a small bowl and add the sugar. Stir until the sugar is dissolved. Sprinkle the yeast over the water and leave for 10 minutes. Combine the flour and salt in a large bowl. Make a well in the flour and add the yeast mixture. Mix thoroughly. Place the dough in a greased bowl, cover and store in a warm place for 30 minutes or until the dough has risen. Knead for another 2 minutes. Divide the mixture in half, place a dead mouse in the base of each loaf, and place on a baking tray. Wait 10 more minutes for the dough to rise again and bake at 200 degrees for 40 minutes. The bread should be golden brown and should sound hollow if you tap the crust.

33 blocks

16

How many 10 minute blocks?

Look at the following program which evaluates the number of completed ten minute blocks given the number of hours and minutes.

```
public class TenMinuteBlocks {
    public void start() {
        int hours = 5;
        int minutes = 34;
        int totalMins, tenMinBlocks;

        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;
        System.out.println(tenMinBlocks + " blocks");
    }
}
```

Repetition!

17

The problem - Do calculation **twice** ..

```
public class TenMinuteBlocks {
    public void start() {
        int hours = 5;
        int minutes = 34;
        int totalMins, tenMinBlocks;

        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;
        System.out.println(tenMinBlocks + " blocks");

        hours = 11;
        minutes = 6;
        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;
        System.out.println(tenMinBlocks + " blocks");
    }
}
```

33 blocks

66 blocks

Repetition!

18

The problem - Do calculation **three times** ....

```
public class TenMinuteBlocks {
    public void start() {
        int hours = 5;
        int minutes = 34;
        int totalMins, tenMinBlocks;
        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;
        System.out.println(tenMinBlocks + " blocks");
        hours = 11;
        minutes = 6;
        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;
        System.out.println(tenMinBlocks + " blocks");
        hours = 2;
        minutes = 16;
        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;
        System.out.println(tenMinBlocks + " blocks");
    }
}
```

33 blocks

66 blocks

13 blocks

19

The solution

Making bread dough

Put half of the water in a small bowl and add the sugar. Stir until the sugar is dissolved. Sprinkle the yeast over the water and leave for 10 minutes. Combine the flour and salt in a large bowl. Make a well in the flour and add the yeast mixture. Mix thoroughly. Place the dough in a greased bowl, cover and store in a warm place for 30 minutes or until the dough has risen. Knead for another 2 minutes.

Baking bread dough

Wait 10 more minutes for the dough to rise again and bake at 200 degrees for 40 minutes. The bread should be golden brown and should sound hollow if you tap the crust.

Recipe 1: Rectangular loaf

Make the bread dough. Divide the mixture in half and put each half into a rectangular baking tin. Bake the bread dough.

Recipe 2: Round loaf

Make the bread dough. Divide the mixture in half. Shape each half into a ball and place on a baking tray. Bake the bread dough.

20

The solution

Identify sections of code that needs to be executed repeatedly:

```
public class TenMinuteBlocks {
    public void start() {
        int hours = 5;
        int minutes = 34;
        int totalMins, tenMinBlocks;

        totalMins = hours * 60 + minutes;
        tenMinBlocks = totalMins/10;

        System.out.println(tenMinBlocks + " blocks");
    }
}
```

then **define that code in a method**, and **call the method** whenever we need the code executed.

21

Defining methods

A **method** is a self-contained section of code for accomplishing a task. We can think of a method as a mini-program.

The syntax for defining a method is

```
private returnType methodName(Formal Parameters){

    block of java statements
}
```

22

Method signature, header

An example of a method is:

```
private int getBlocks(int hrs, int mins) {
      int totalMins, blocks;
      totalMins = hrs * 60 + mins;
      blocks = totalMins/10;
      return blocks;
}
```

The **signature** of the method is the name and the list of parameters, for example

```
getBlocks(int hrs, int mins)
```

The **header** of the method is the name and the list of parameters, for example

```
private int getBlocks(int hrs, int mins)
```

23

Parameters

- Parameters are used to pass information to a method
- Parameters in the method are initialised when the method is called
- Each parameter must have a type
- Each parameter in the list is separated by a comma

Example

```
private int getBlocks(int hrs, int mins) {
      int totalMins, blocks10;
      totalMins = hrs * 60 + mins;
      blocks10 = totalMins/10;
      return blocks10;
}
```

List of parameters

method definition

24

Calling a method

To execute the code in a method, the method needs to be called.

To call a method you use the name of the method followed by the list of values which you wish to pass to the method, for example

```
private int getBlocks(int hrs, int mins) {
      int totalMins, blocks;
      totalMins = hrs * 60 + mins;
      blocks = totalMins/10;
      return blocks;
}
```

method definition

```
getBlocks(5, 34);
```
method call

```
getBlocks(2, 48);
```
A second method call

```
getBlocks(1, 55);
```
A third method call

Parameters

25

```
Passing parameters is like
initialising the variables:
      num = x;
      ave = 4.5;
      message = "Testing";
```

Passing parameters is just like assigning values to the variables declared in method signature. [The parameters passed must have same type of parameters and must be in the same order.]

```
getString(x, 4.5, "Testing");
```

```
private String getString(int num, double ave, String message) {
      //add code
}
```

```
private String getInfo(String name, int age, boolean isSingle){
```

Ex01 - parameter list

The following is a call to the getInfo() method:

```
String result = getInfo("Mia", 21, true);
```

Complete the method header for the getInfo() method:

```
private String getInfo(                                    ) {
      //add code
}
```

27

Returning values

This allows a method to pass information back to the invoking code.

The method header must state the type of value it is returning.

A method can only **return** a single value (primitive or object).

```
private int getBlocks(int hrs, int mins) {
      int totalMins, blocks;
      totalMins = hrs * 60 + mins;
      blocks = totalMins/10;
      return blocks;
}
```

28

Return statement

Every method that returns a value must use the keyword "**return**" as the **last** statement in the method.

The value returned must be the **same type** as declared in the method header. For example

```
private int getBlocks(int hrs, int mins) {
      int totalMins = hrs * 60 + mins;
      int blocks = totalMins/10;
      return blocks;
}
```

```
private String getName() {
      return "Fred Fish";
}
```

29

Returning values

When a method call is executed then the returned value **is used in place of** the method call

```
public class CallingMethods {
      public void start() {
            int numBlocks = getBlocks(3, 5);

            numBlocks = getBlocks(6, 35);
      }
      private int getBlocks(int hrs, int mins) {
            int totalMins, blocks;
            totalMins = hrs * 60 + mins;
            blocks = totalMins/10;
            return blocks;
      }
}
```

18

39

30

Returning values

The variable on the left hand side of the method call must be of the same type as the return type of the method.

```
public class CallingMethods {
      public void start() {
            int blocks1 = getBlocks(3, 5);

            int blocks2 = getBlocks(6, 35);
      }
      private int getBlocks(int hrs, int mins) {
            int totalMins, blocks;
            totalMins = hrs * 60 + mins;
            blocks = totalMins/10;
            return blocks;
      }
}
```

31

Ex02

(a), (c) are correct

Which of the following calls to the calculate() method will compile?

```
public class Test02 {
      public void start() {
          double a = calculate(5 , 4);          (a)
          double d = calculate(4.0);            (b)
          int c = calculate(2 , 4.2);           (c)
          int x = calculate(3.5, 1);            (d)
          boolean b = calculate(3, 1.1);        (e)
      }
     private int calculate(int a,double b) {
           int amount = (int)(a + b *2);
           return amount;
      }
}
```

32

Ex03

(a), (e) are correct

Which of the following are correct calls to the lots() method?

```
public class Test03 {
    public void start() {
        String result;
        result = lots(2 + 1, 3, "3");       (a)
        result = lots(2 + 1, 3, 4);         (b)
        result = lots(22, "Hi", 4);         (c)
        result = lots(8, 3.4, 'x');         (d)
        result = lots(22, 1.2, "4");        (e)
    }
    private String lots(int a, double b, String c) {
          System.out.println(a + b + c);
          return c + " " + a;
}   }
```

33

Ex04

The returnType and the type of value returned do not match.

What is incorrect with the following method definition?

```
public class Problem1 {
       public void start() {
              int b = calculate(3, 1.1);
       }
       private int calculate(int a, double b) {
              double amount = a + b *2;
              return amount;
       }
}
```

34

Ex05

The return statement must be the last statement in the method

What is incorrect with the following method definition?

```
public class Problem2 {
       public void start() {
              int b = calculate(3, 1.1);
       }
       private int calculate(int a, double b) {
              int amount = (int)(a + b * 2);
              return amount;
              amount = amount + 2;
       }
}
```

35

Ex06 - Give the output

2241: $150

```
public class ProcessUserInput {
  public void start() {
     int userId = getValueFromUser("User id: ");
     int pin = getValueFromUser("Pin number: ");
     int amount = getValueFromUser("Amount: ");
    System.out.println(userId + ": $" + amount);
  }
  private int getValueFromUser(String prompt) {
       System.out.print(prompt);
       String input = Keyboard.readInput();
       int value = Integer.parseInt(input);
       return value;
  }
}
```

The user enters the values 3341, 5568 and 150 when prompted

36

Ex08 - Give the output

NZ$100 = AU$85.0

NZ$200 = Euros 100

```
public class ProcessUserInput {
    public void start() {
      String outCurrency = convertAmt("AU$ ", 100, 0.85);
      System.out.println("NZ$100 = " + outCurrency );

      outCurrency = convertAmt("Euros ", 200, 0.5);
      System.out.println("NZ$200 = " + outCurrency );
    }

    private String convertAmt(String currency, int amt,
                                          double perDollar) {
       double amount = amt * perDollar;
       String currString = currency + amount;
       return currString ;
    }
}
```

37

Reasons for defining methods

Avoiding repetition of code is one important reason for using methods.

There are other good reasons:

- makes the code much clearer than a single, very long, `start()` method
- makes programs easier to develop by clearly identifying individual tasks

Each method should represent a single task

38

What you need to know

Defining methods: list of parameters, the returnType, the return statement

Calling a method: the parameters must match in order and type, the method call is replaced by the value which is returned by the method, the variable to which the method call is assigned must be of the same type as the return type of the method