Advanced Programming Techniques in Java

Objectives

- Java Syntax Overview

Review:

- In the Java programming langua
- A program is made up of one or more *classes*
- A class contains one or more *methods*
- A method contains program *statements*

- A Java application always contains a meth
  `main`

Review: Obje

- Classes and Objects
- Class definitions in .java files
- Def: a *class* is a named description for a group of entities
- *Objects* or *instances* of the class is the group of entities
- The characteristics are the attributes (*data fields*) for eac can be performed on these objects

Review: hello!

```
public class Hello{ public static void
        main(String[] args){
                System.out.println("Hello Worl
        }
}
```

- Everything in Java must be inside a class
- Every file may only contain one public class
- The name of the file must be the name of the c extension
- Thus, **`Hello.java`** must contain one public

Form

- Syntax

  ```
  System.out.printf("format string", <li
  ```

- The *format string* is like placeholders where the
- These placeholders are used instead of + concatenation
- %d integer
- %f real numbers
- %s string
- Example

```
int       x = 3; int y = -17;
System.out.printf("x is %d and y is %d
```

Note: `printf()` does not drop to the next line unless you use

`printf` precision

`%.Df` real number, rounded to **D** digits after de

`%W.Df` real number, **W** characters wide, **D** digit

```
double gpa = 3.253764;
System.out.printf("your GPA is %.1f\r
System.out.printf("more precisely: %8
```

Output

```
your
GPA is 3.3 more
precisely:   3.254
```

printf with St

A simple string	`printf("'%s'", "He`
A string with a minimum length	`printf("'%10s'", "`

Minimum length, left-justified	`printf("'%-10s'",`

Variables and D

Variables

- A **variable** is a name for a location in memory
- It can be thought of as a container which holds values f

- A variable must be declared by specifying the the type of information that it will hold

```
int total;
```

data type va

Variables

- In order to use a variable in a program you t
- Variable Declaration
- Variable Initialization

- A variable can be given an initial value in the

```
int total =
```

Assignment

- An *assignment statement* changes the value o

```
total = 35;
```

- The value that was originally in `total` is ove
- You can assign only a value to a variable that i declared type

Data Types

- Data types classify the different values to

  In Java there are two types of data types:

- Primitive Data Types

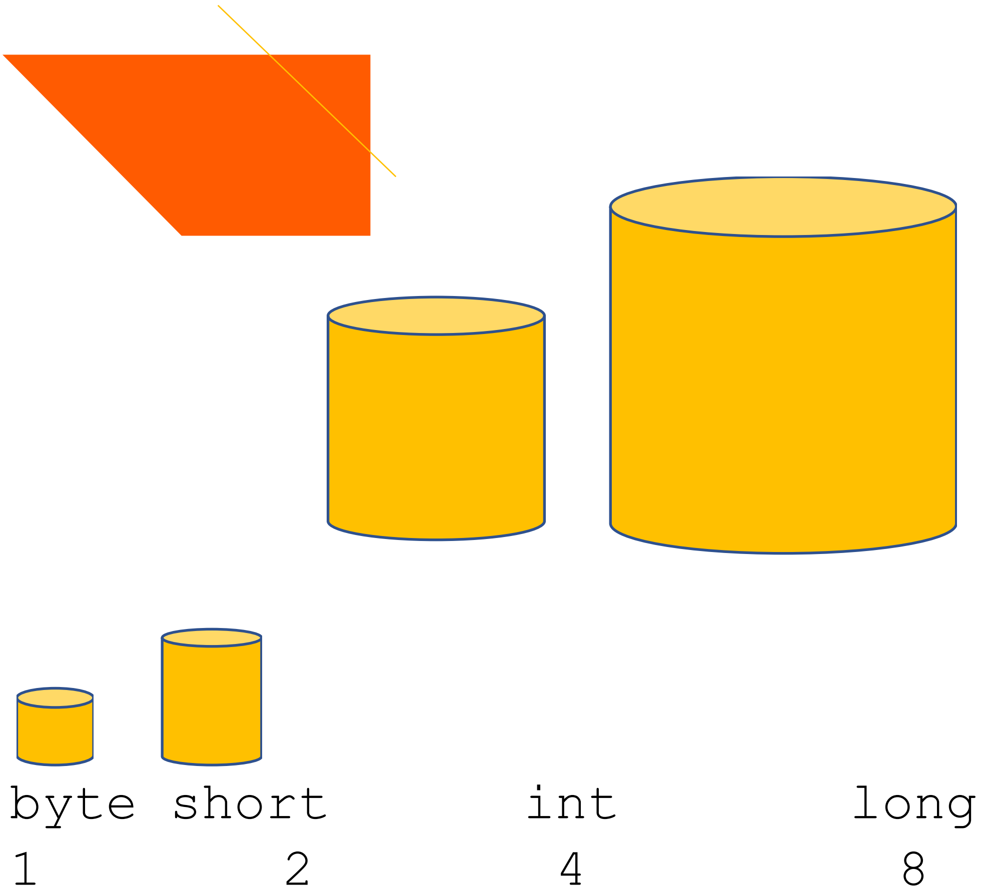
- Non-primitive Data Types

Primitive Data Types

- Primitive Data Types are predefined and available within the Java language
- There are 8 primitive types: `byte, short,` `double, and boolean`

B

bo



Data type
byte
short
int
long
float

do

bo

cha

Primitive Data Typ

- byte -128 to 127
- short -32,768 to 32,767
- int -2,147,483,648 to 2,14
- long -9,223,372,036,854,77
- float ±10^{38} incl. 0 with 6 digi
- double ±10^{308} incl. 0 with 1
- char Unicode character set
- boolean true, false

Example

```
public class ChangeAdder { public static v
      main(String[] args){
            int quarters =
            10; int dimes =
            3; int nickels =
            7; int pennies =
            6; int change =
            0;
            change = 25*quarters+10*dimes
            System.out.println("total in
      }
}
```

Java bas

Data Conversions

- Sometimes it is convenient to convert data f
- For example, we may want to treat an integ
  during a computation
- Conversions must be handled carefully to av
  Data conversions can occur in three ways:
- Assignment conversion
- Arithmetic promotion
- Casting

Data Conversions

- *Assignment conversion* occurs when a value of
  another
- *Arithmetic promotion* happens automatically w
  convert their operands
- *Casting* is accomplished by explicitly casting a v
- To cast, the type is put in parentheses in front of the va
- For example, if `total` and `count` are integers, but we war
  can cast `total`:

```
result = (double) total / count;
```

Type cast operator

Operators

- Operators are symbols that perform opera

Op
*, /, %
+, −
==
!=
<, >
<=, >=
& &
||
!

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Unary Operators
- Assignment Operators

++, --, –
(type)
=, +=, -=,

String Concate

- This operator combines several strings into a sin
  other data into a new longer string

- Example:

```
System.out.println("Grade: " + (95.1 +
```

- Output:

```
Grade: 83.5
```

Class Libraries

- A *class library* is a collection of classes that we
- The `System` class, the `Scanner` class, and t standard class library
- Related classes are grouped into packages

Package	Purpose
java.lang	General support
java.applet	Creating applets for
java.awt	Graphics and graph
java.util	Utilities
…	…

Interactive Progra

- The `Scanner` class is used to get input from interactive
- It is part of the `java.util` package

- A `Scanner` object can read input from man
- The console window (`System.in`)
- Files, web sites, databases, …

The `import` Dec

- In order to access a package, you need to inclu

- You can *import* the class, and then use just t
  `java.util.Scanner;`
- To import all classes in a particular package, you can use
  `java.util.*;`

- All classes of the `java.lang` package (e.g., imported automatically into all programs

`Scanner` class

- First a `Scanner` object is created

```
Scanner <variable-name> = new
```

This parameter
to read from the

- Example: `Scanner console = new Sc`
- Then various methods can be used to read di keyboard
- Example: `int num = scan.nextInt(`

Scanner metho

Method	D
`nextInt()`	reads an `int` from the
`nextDouble()`	reads a `double` from t

`next()`	reads a one-word `Stri`
`nextLine()`	reads a one-*line* `Strin`

```
Scanner console = new Scanner(System.in)
System.out.print("How old are you? ");
int age = console.nextInt();
System.out.println("You typed " + age);
```

Example

```
import java.util.*;
public class UserInputExample {
   public static void main(String[] args
      Scanner console = new Scanner(Syst
      System.out.println("How old are yo
      int age = console.nextInt();
      int years = 65 - age;
      System.out.println(years + " years
   }
}
```

- Console window:

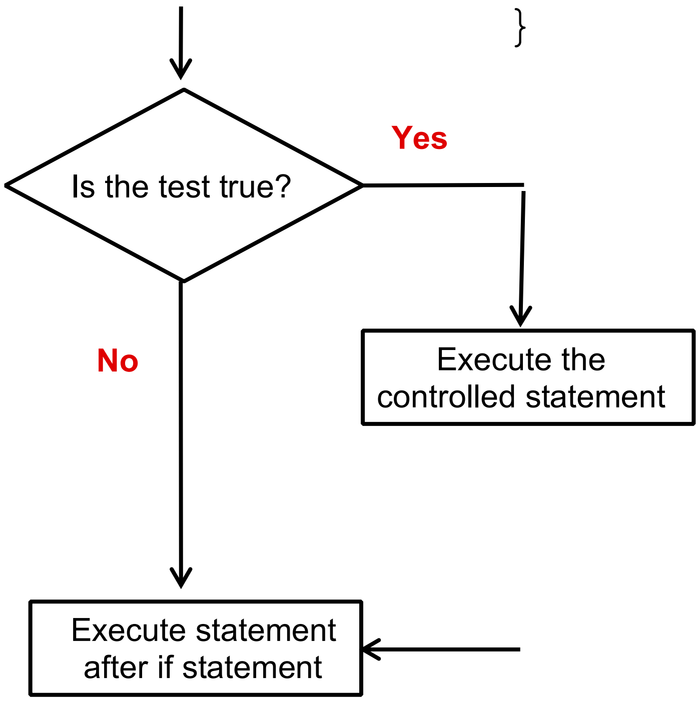
```
How old are you?   29
36 years to retirement!
```

```
if
    s
}els



    s
}
```

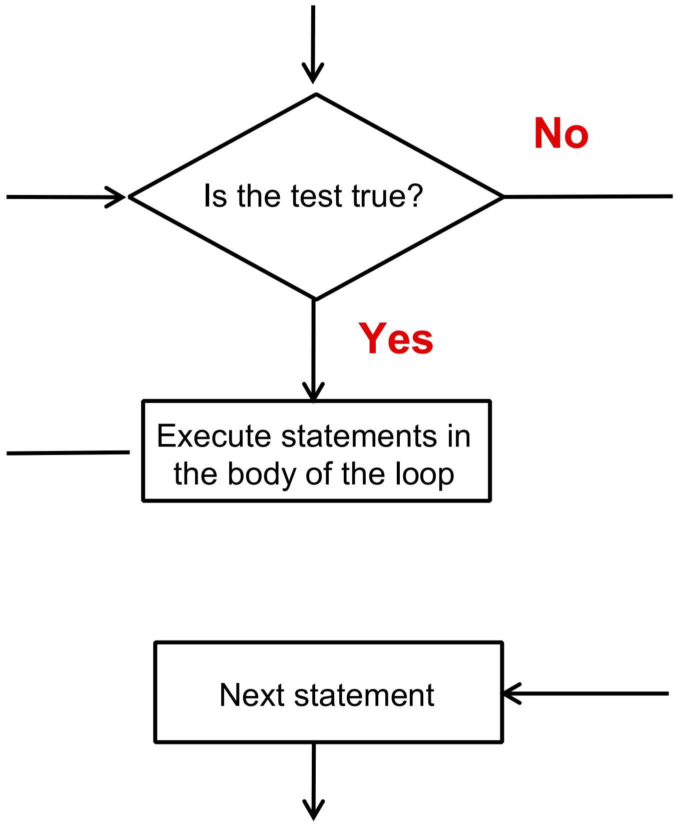
Conditional Statements

- A *conditional statement* lets us choose which statement will be executed next

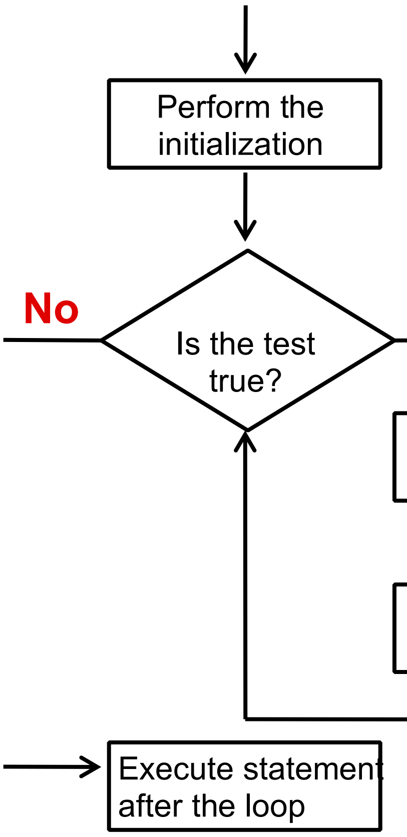
```
if (cond
    state
}
```


Is the test true?
Yes
No
Execute the controlled statement
Execute statement after if statement

Repetition Statem



```
while (condition){
    statement(s);
}
```

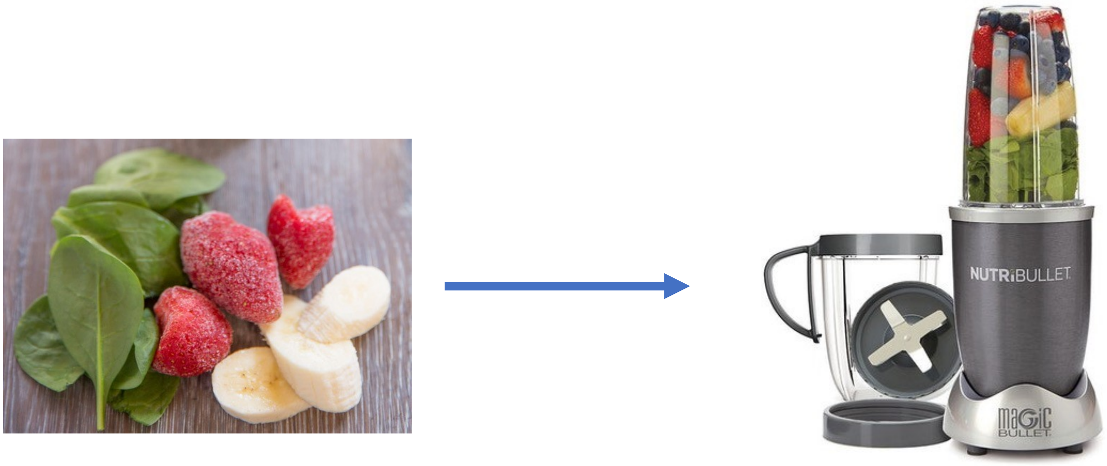


```
for (initialization; cond
    statement(s);
}
```

Metho

Methods

- A program that provides some functionality can
  statements
- A method groups a sequence of statements and
  easyto-understand functionality
- A method can take input, perform actions, and

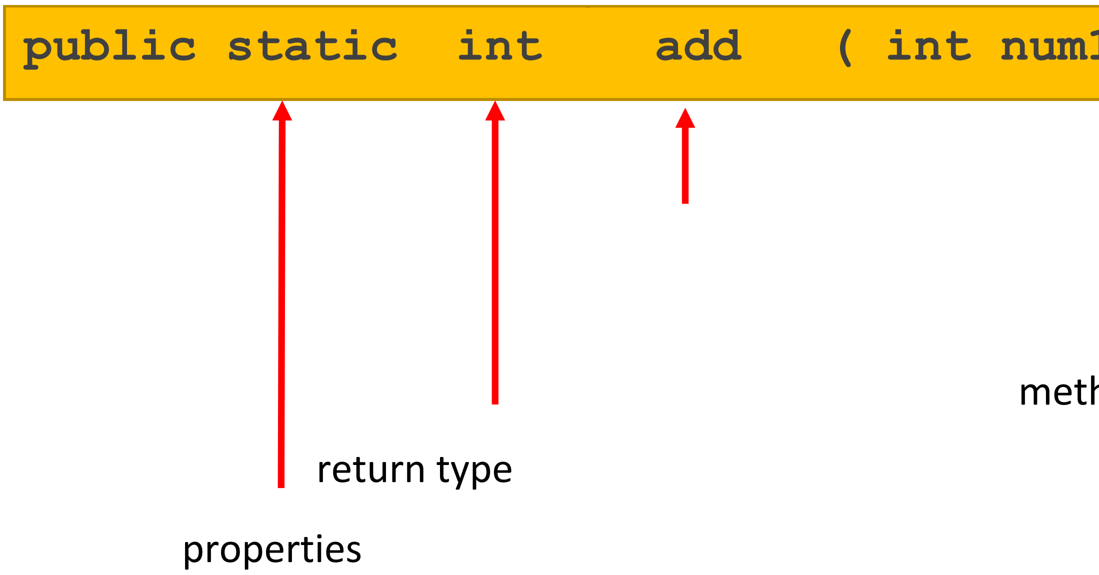


Method Declaratio

- A *method declaration* specifies the code that wil invoked (or called)
- When a method is invoked, the flow of control ju its code
- When complete, the flow returns to the place wl continues
- The invocation may or may not return a value, d defined

Method Header

- A method declaration begins with a *method he*



```
public static  int    add    ( int num1
```

meth

return type

properties

- The parameter list specifies the type and name of each p
- The name of a parameter in the method declaration is ca
- **static** indicates a *static* or an *object/instance* method
- A method that is not static, is an instance method

Static Vs

- Static methods
- There is one per class
- Instance methods
- There is one per object of the class ▪ Static metho

methods

```
public class Car{
…
?? float km2Miles(floa
?? float getOdometerMi
```

}

Java Constants

```
public static final int ARRAY_SIZE =

private static final String URL = "ts
```

- Constants in Java have to be initialized wher
- After that, they are read only.

Method Body

- A method header is followed by the *method b*

```
public static int add( int num1, int n
        int result = 0;
        result = num1 + num2;
        return result;
}
```

return statement

- The return expression must be consistent with
- The variable result is a local variable. It is creat and is destroyed when it finishes executing

The return stat

- The return statement sends out a value as th

```
return expressi
```

- The *return type* of a method indicates the type o to the calling location
- A method that does not return a value has a vo

Parametrization

- A *parameter* is a special type of variable that a method
- A method can accept multiple parameters (se
- Each time a method is called, the *actual para* into the formal

Declaration syntax

```
public static  int    add    ( int num1, in
```

Call syntax

formal para

```
add (5, 9);
```

actual parameters