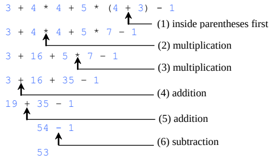
CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Expressions

Remember operator precedence and associativity:



```
3 + 4 * 4 + 5 * (4 + 3) - 1
                   (1) inside parentheses first
3 + 4 * 4 + 5 * 7 - 1
                   (2) multiplication
3 + 16 + 5 * 7 - 1
                   (3) multiplication
3 + 16 + 35 - 1
                   (4) addition
19 + 35 - 1
                   (5) addition
    54 - 1
                   (6) subtraction
    53
```

Java Logical and Arithmetic Operator Precedence Rules

1. **!** - (unary)
2. ***** / **%**
3. + -
4. < <= > >=
5. == **!=**
6. ^ **&** |
7. **&&**
8. ||

Division /and %

5 / 2 yields an integer, which?

5.0 / 2 yields a double, which?

5 % 2 yields the integer remainder of the division, which?

p = (p/q)*q + p%q

Conversion Rules

When performing a binary operation involving two operands of different types, Java automatically converts the operand; promotes to wider type:

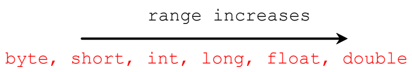
1. If one of the operands is double, the other is converted into double.
2. Otherwise, if one of the operands is float, the other is converted into float.
3. Otherwise, if one of the operands is long, the other is converted into long.
4. Otherwise, both operands are converted into int.

Type Casting

Implicit casting
  `double d = 3;` (type widening)

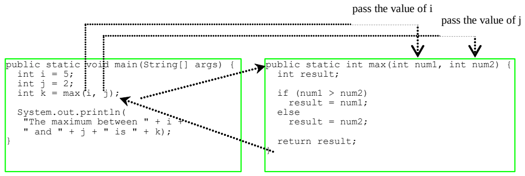
Explicit casting for narrowing
  `int i = (int)3.0;` (type narrowing)
  `int i = (int)3.9;` (Fraction part is truncated)

What is wrong? int x = 5 / 2.0;


range increases

byte, short, int, long, float, double

Calling Methods



pass the value of i

pass the value of j

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

?

CS1 Review

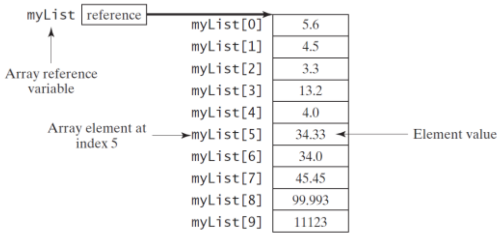
CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Arrays

Array is a data structure that represents a collection of the same types of data.

```
double[] myList = new double[10];
```



Passing Arrays to Methods

```
public static void printArray(int[] array) {
  for (int i = 0; i < array.length; i++) {
    System.out.print(array[i] + " ");
  }
}
```

```
Invoke the method

int[] list = {3, 1, 2, 6, 4, 2};
printArray(list);
```

```
Invoke the method
printArray(new int[]{3, 1, 2, 6, 4, 2});
```

Anonymous array

Linear Search

The linear search approach compares the key element, key, *sequentially* with each element in the array list. The method continues to do so until the key matches an element in the list or the list is exhausted without a match being found. If a match is made, the linear search returns the index of the element in the array that matches the key. If no match is found, the search returns -1.

animation

Linear Search Animation

Key	List							
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8
3	6	4	1	9	7	3	2	8

Binary Search

Consider the middle element in a sorted array:

- If the key is less than the middle element, you only need to search the key in the first half of the array.
- If the key is equal to the middle element, the search ends with a match.
- If the key is greater than the middle element, you only need to search the key in the second half of the array.

animation

Binary Search

Key	List							
8	1	2	3	4	6	7	8	9
8	1	2	3	4	6	7	8	9
8	1	2	3	4	6	7	8	9

Two-dimensional Arrays

```
int[][] array = {
  {1, 2},
  {3, 4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

- what is array[2]?
- what are the array bounds ? (there are many)

array.length ?

array[0].length ?

array[1].length ?

array[2].length ?

array[3].length ?

?

CS1 Review

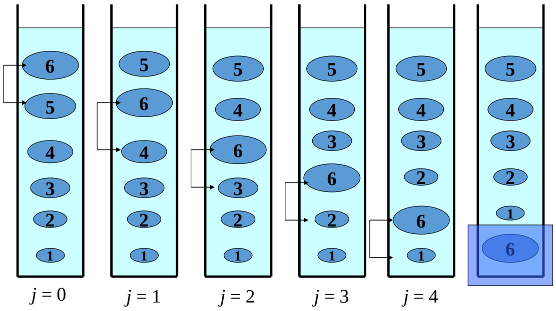
CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

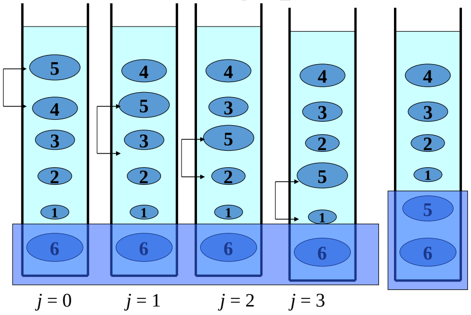
Bubble Sort

- Compares neighboring elements, and swaps them if they are not in order
  - Effect: the largest value will “bubble” to the last position in the array.
  - Repeating the process will bubble the 2nd to largest value to the 2nd to last position in the array

Bubble sort (First pass)
i = 0



Bubble sort (Second pass)
i = 1



Bubble Sort

```
public void bubbleSort (Comparable [] array) {
   for (int position = array.length-1; position>=0;
        position--) {
        for (int i = 0 ; i < position; i++) {
                if (array[i].compareTo(array[i+1]) > 0)
                        swap(array, i, i+1);
        }
   }
}
```

Inner Invariant: array[i] is the largest element in the first i elements in the array

Outer Invariant: After i iterations the largest i elements are sorted

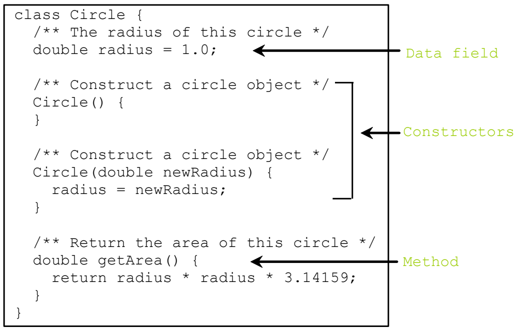
?

CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Classes



```
class Circle {
  /** The radius of this circle */
  double radius = 1.0;

  /** Construct a circle object */
  Circle() {
  }

  /** Construct a circle object */
  Circle(double newRadius) {
    radius = newRadius;
  }

  /** Return the area of this circle */
  double getArea() {
    return radius * radius * 3.14159;
  }
}
```

Constructors

```
Circle() {
}

Circle(double newRadius) {
  radius = newRadius;
}
```

Constructors are a special kind of methods that are invoked to construct objects.

Constructors, cont.

A constructor with no parameters is referred to as a *no-arg constructor*.

Constructors must have the same name as the class.

Constructors do not have a return type—not even void.

Constructors are invoked using the new operator. When an object is created, constructors play the role of initializing objects.

Default Constructor

A class may be defined without constructors. In this case, a no-arg constructor with an empty body is implicitly defined in the class. This constructor, called *a default constructor*, is provided automatically *only if no constructors are explicitly defined in the class*.

Instance Variables, and Methods

Instance variables belong to a specific instance.

Instance methods are invoked by an instance of the class.

Instance variables and methods are specified by omitting the **static** keyword.

Static Variables, Constants, and Methods

Static variables are shared by all the instances of the class.

Static methods are not tied to a specific object.

Static constants are final variables shared by all the instances of the class.

To declare static variables, constants, and methods, use the **static** modifier.

Visibility Modifiers and Accessor/Mutator Methods

By default, the class, variable, or method can be accessed by any class in the same package.

- `public`

  The class, data, or method is visible to any class in any package.

- `private`

  The data or methods can be accessed only by the declaring class.

The get and set methods are used to read and modify private properties.

The this Keyword

- The this keyword is the name of a reference that refers to an object itself. One common use of the this keyword is reference a class's *hidden data fields*.
- Another common use of the this keyword to enable a constructor to invoke another constructor of the same class.

Calling Overloaded Constructor

```
public class Circle {
  private double radius;

  public Circle(double radius) {
    this.radius = radius;
  }

  public Circle() {
    this(1.0);
  }

  public double getArea() {
    return this.radius * this.radius * Math.PI;
  }
}
```

this must be explicitly used to reference the data field radius of the object being constructed

this is used to invoke another constructor

Every instance variable belongs to an instance represented by this, which is normally omitted

?

CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Basic Component: Class

A Class is a software bundle of **related states** (**properties**, or **variables**) and **behaviors** (**methods**)

- State is stored in instance variables
- Method exposes behavior

Basic Components

- **Class**: ***Blueprint*** from which objects are created
  - Multiple Object Instances created from a class
- **Interface**: A ***Contract*** between classes and the outside world.
  - When a class **implements** an interface, it **promises to provide the behavior** published by that interface.
- **Package**: a ***namespace*** (directory) for organizing classes and interfaces

Data Encapsulation

- An ability of an object **to be a container** (or capsule) for related properties and methods.
  - Preventing unexpected change or reuse of the content
- **Data hiding**
  - Object can shield variables from external access.
    - Private variables
    - Public **accessor** and **mutator** methods, with potentially limited capacities, e.g. only read access, or write only valid data.

Data Encapsulation

```
public class Clock{
  private long time, alarm_time;

  public void setTime(long time){
      this.time = time;
  }
  public void setAlarmTime(long time){
      this.alarm_time = time;
  }
  public long getTime(){return time}
  public long getAlarmTime(){return alarm_time}
  public void noticeAlarm(){ … //ring alarm }
  }
}
```

?

CS1 Review

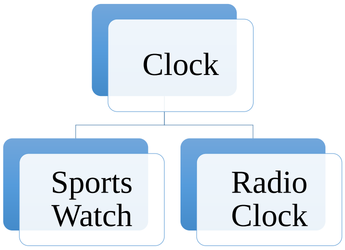
CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Inheritance

- The ability of a class to ***derive*** properties and behaviors from a previously defined class.
- **Relationship** among classes.
- Enables **reuse** of software components
  – e.g., java.lang.Object()
  – toString(), equals(), etc.

Example: Inheritance


Clock

Sports Watch

Radio Clock

Example: Inheritance – cont.

```
Public class SportsWatch extends Clock {
   private long start_time;
   private long end_time;

   public long getDuration()
   {
     return end_time - start_time;
   }

}
```

Overriding Methods

```
public class RadioClock extends Clock
  {
  @override
    public void noticeAlarm(){
            ~~ring alarm~~
            turn_on_the_Radio
     }
}
```

?

CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Are superclass's Constructor Inherited?

No. They are not inherited.

They are invoked explicitly or implicitly.

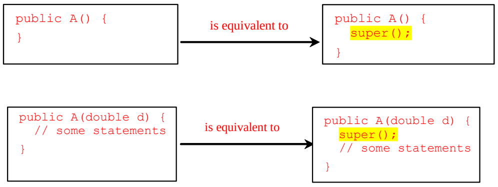
Explicitly using the super keyword.

A constructor is used to construct an instance of a class. Unlike data and methods, a superclass's constructors are not inherited in the subclass. They can only be invoked from the subclasses' constructors, using the keyword super.

If the keyword super is not explicitly used, the superclass's no-arg constructor is automatically invoked.

Superclass's Constructor Is Always Invoked

A constructor may invoke an overloaded constructor or its superclass's constructor. If none of them is invoked explicitly, the compiler puts super() as the first statement in the constructor. For example,



```
public A() {
}
```

is equivalent to

```
public A() {
  super();
}
```

```
public A(double d) {
  // some statements
}
```

is equivalent to

```
public A(double d) {
  super();
  // some statements
}
```

Using the Keyword super

The keyword super refers to the superclass of the class in which super appears. This keyword can be used in two ways:

- To call a superclass constructor
- To call a superclass method

CAUTION

You must use the keyword super to call the superclass constructor, instead of the superclass constructor's name.

Java requires that the constructor call super appear first in the constructor.

Example on the Impact of a Superclass without no-arg Constructor

Error: Fruit has no no-arg constructor, so Apple fails:

```
public class Apple extends Fruit {
}

class Fruit {
  public Fruit(String name) {
    System.out.println("Fruit's constructor is invoked");
  }
}
```

?

CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Defining a Subclass

A subclass inherits methods and data from a superclass. You can also:

- Add new properties
- Add new methods
- Override the methods of the superclass

Overriding vs. Overloading

Overloading occurs when two or more methods **in the same class** have the same method name but different parameters.

Overriding means having two methods with the same method name and parameters (i.e., method signature). One of the methods is in **the parent class** and the other is in **the child class.**

Overriding Methods in the Superclass

A subclass inherits methods from a superclass. Sometimes it is necessary for the subclass to modify the implementation of a method defined in the superclass. This is referred to as *method overriding*.

```
public class Circle extends GeometricObject {
  // Other methods are omitted


  /** Override the toString method defined in GeometricObject */
  public String toString() {
    return super.toString() + "\nradius is " + radius;
  }
}
```

NOTE

An instance method can be overridden only if it is accessible.

Thus a private method cannot be overridden, because it is not accessible outside its own class.

If a method defined in a subclass is private in its superclass, the two methods are completely unrelated.

The Object Class and Its Methods

Every class in Java is descended from the java.lang.Object class. If no inheritance is specified when a class is defined, the superclass of the class is Object.

```
public class Circle {
  ...
}
```

Equivalent

```
public class Circle extends Object {
  ...
}
```

The toString() method in Object

The toString() method returns a string representation of the object. The default implementation returns a string consisting of a class name of which the object is an instance, the at sign (@), and a number representing this object.

```
Loan = new Loan();
System.out.println(loan.toString());
```

The code displays something like Loan@15037e5 . This message is not very helpful or informative. Usually you should override the toString method so that it returns a digestible string representation of the object.

?

CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

Polymorphism

Polymorphism means that a variable of a supertype can refer to a subtype object.

A class defines a type. A type defined by a subclass is called a *subtype*, and a type defined by its superclass is called a *supertype*.

Therefore, you can say that **Circle** is a subtype of **Shape** and **Shape** is a supertype for **Circle**.

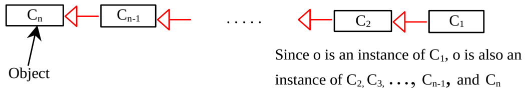
Dynamic Binding

Object o is an instance of class C_1

C_1 is a subclass of C_2, C_2 is a subclass of C_3, ...etc.,

C_n is the Object class.

If a method in o invokes a method p, the JVM searches the implementation for the method p in C_1, C_2, ..., C_{n-1} and C_n, in this order, until it is found, and that method is invoked.



Since o is an instance of C_1, o is also an instance of C_2, C_3, . . . , C_{n-1}, and C_n

Casting from Superclass to Subclass

Explicit casting must be used when casting an object from a superclass to a subclass. This type of casting may not always succeed.

```
Apple x = (Apple)fruit;

Orange x = (Orange)fruit;
```

The `instanceof` Operator

Use the `instanceof` operator to test whether an object is an instance of a class:

```
Object myObject = new Circle();
... // Some lines of code
/** Perform casting if myObject is an instance of
  Circle */
if (myObject instanceof Circle) {
  System.out.println("The circle diameter is " +
    ((Circle)myObject).getDiameter());
  ...
}
```

The `equals` Method

The `equals()` method compares the contents of two objects. The default implementation of the equals method in the Object class is as follows:

```
public boolean equals(Object obj) {
  return this == obj;
}
```

For example, the equals method is overridden in the Circle class.

```
public boolean equals(Object o) {
  if (o instanceof Circle) {
    return radius == ((Circle)o).radius;
  }
  else
    return false;
}
```

NOTE

The == comparison operator is used for comparing two primitive data type values or for determining whether two objects have the same references.

The equals method is intended to test whether two objects have the same contents, provided that the equals method is overriden in the defining class of the objects.

?

CS1 Review

CS2: Data Structures and Algorithms
Colorado State University

Chris Wilcox, Russ Wakefield, Wim Bohm,
Dave Matthews, Sudipto Ghosh, Benjamin Say

What is an abstract class?

- A class that is declared abstract
- May (or may not) contain abstract methods
  - Method declarations without implementations
- We use abstract methods when the super class has no way to implement the method, e.g. area() for geometricObject (completely different for circle and rectangle).

Abstract method

- Must be contained in an abstract class.
- A concrete subclass of a superclass must implement all the abstract methods,
- otherwise, the subclass must be defined abstract.

Abstract class constructors

- An abstract class cannot be instantiated using the new operator.
- However, constructors can be defined to initialize the instance variables of the abstract class
- Constructors of a subclass invoke the constructors of the super class.
- The constructors of GeometricObject are invoked in the Circle class and the Rectangle class.

Superclass of abstract class may be concrete

- A subclass can be abstract even if its superclass is concrete.
- New abstract methods can be declared in the subclass.
- For example, the Object class is concrete, but its subclasses, such as GeometricObject, may be abstract.

Concrete method overridden to be abstract

- Rarely done.
- Useful when the implementation of the method in the superclass becomes invalid in the subclass.
- A subclass can override this method from its superclass to define it abstract.
- The subclass must be defined abstract.

Using abstract class as type

- An abstract class can be used as a data type.
- The below statement correctly creates an array whose elements are of GeometricObject type.

```
GeometricObject[] geo = new GeometricObject[10];
```

- Note that the array elements are references to GeometricObject instances. No GeometricObjects are instantiated.

Cannot write: geo[0]= new GeometricObject();

?