1.00/1.001
Spring 2012
Quiz 1 Review

Course Topics Overview

1. **Control and scope**
2. **Classes and objects** — *Quiz 1*
3. **Arrays, ArrayLists**
4. Recursion
5. Inheritance — *Quiz 2*
6. Graphical user interfaces
7. Numerical methods
8. Input/output streams — *Final*
9. Sensors and threads
10. Data structures

1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching (if/else)
   - Iteration (while/do while/for)
4. Methods
   - Argument passing
   - Variable scope

1. Control and Scope

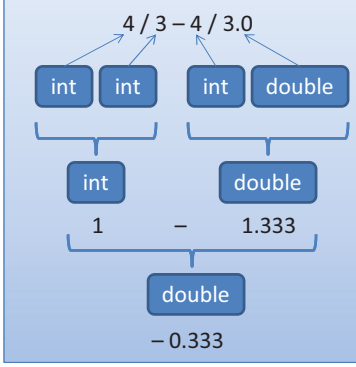
1. ***Data types***
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching (if/else)
   - Iteration (while/do while/for)
4. Methods

Type	Size (bits)	Range
boolean	1	true or false
char	16	ISO Unicode character set
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2,147,483,648 to 2,147,483,647
long	64	-9,223,372,036,854,775,808L to 9,223,372,036,854,775,807L
float	32	1.4E-45F to 3.4E+38F (6-7 significant digits)
double	64	4.9E-324 to 1.8E+308 (15 significant digits)

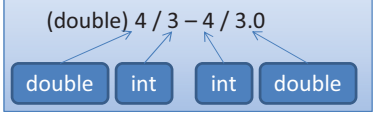
1. Control and Scope

1. Data types
   - ***Promotion and casting***
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching (if/else)
   - Iteration (while/do while/for)
4. Methods

Promotion example



Casting example



1. Control and Scope

1. Data types
   - Promotion and casting
2. ***Operators***
   - ***Precedence***
   - Numerical problems
3. Control structures
   - Branching (if/else)
   - Iteration (while/do while/for)
4. Methods
   - Argument passing
   - Variable scope

Arithmetic operators

Operators	Meaning	Example	Associativity
++	increment	i= d++; x= ++q;	Right to left
--	decrement	--z; y= (a--) + b;	
+ (unary)	unary +	c= +d;	
- (unary)	unary –	e= -f;	
*	multiplication	a= b * c * d;	Left to right
/	division	e= f / g;	
%	modulo	h= i % j;	
+	addition	k= m + n + p;	Left to right
-	subtraction	q= s – t;	

Assignment operator: =

Boolean operators

Equal	==	Not equal	!=
Less than	<	Less than or equal	<=
Greater than	>	Greater than or equal	>=
Logical and	&&	Logical or	||
Not	!		

1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - ***Numerical problems***
3. Control structures
   - Branching (if/else)
   - Iteration (while/do while/for)
4. Methods
   - Argument passing
   - Variable scope

Common problems

- Integer divide
- Divide by zero
- 0 / 0 = NaN
- Exceeding capacity of data type
- Decimal imprecision and error

1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - ***Branching (if/else)***
   - Iteration (while/do while/for)
4. Methods
   - Argument passing
   - Variable scope

`if / else` example

```
boolean b = 3 > 4;
String s = "hello";

if (b) {
  ...
} else if (s.equals("bye")){
  ...
} else {
  ...
}
```

1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching (if/else)
   - ***Iteration (while/do while/for)***
4. Methods
   - Argument passing
   - Variable scope

`while` example

```
int i = 0;
while (i<10) {
  ...
  i++;
}
```

`for` example

```
for (int i=0; i<10; i++){
  ...
}
```

1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching
   - Iteration
4. ***Methods***

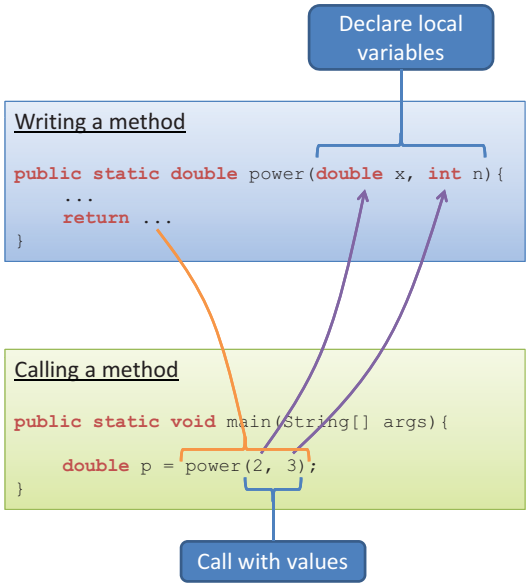
Writing a method

```
public static double power(double x, int n){
}
```

Access | Object or class-based | Return type | Name | Arguments (input)

1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching
   - Iteration
4. Methods
   - ***Argument passing***
   - Variable scope



1. Control and Scope

1. Data types
   - Promotion and casting
2. Operators
   - Precedence
   - Numerical problems
3. Control structures
   - Branching
   - Iteration
4. Methods
   - Argument passing
   - ***Variable scope***

```
public class Scope {
    private int num;

    public Scope(int n) {
        num = n;
    }

    public double calcSomething(int a) {
        double result = 0;
        for (int i = 1; i < a; i++) {
            result += num * i;
        }
        return result;
    }

    public static void main(String[] args) {
        int n = 4;
        Scope s = new Scope(n);
        double d = s.calcSomething((a));
    }
}
```

Scope error!

2. Classes and Objects

1. Making a class
   - Data members
   - Constructor
   - Methods
2. Creating objects
   - Memory allocation
3. Calling methods
   - Passing object arguments
4. Access
5. Static vs. non-static

2. Classes and Objects

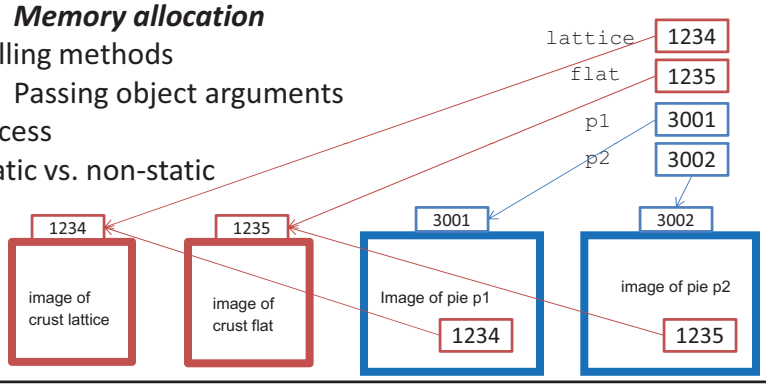
1. ***Making a class***
   - ***Data members***
   - ***Constructor***
   - ***Methods***
2. Creating objects
   - Memory allocation
3. Calling methods
   - Passing object arguments
4. Access
5. Static vs. non-static

```
public class Pie {
   private int apples;
   private double cupsSugar;
   private Crust myCrust;

   public Pie(int a, double s, Crust c){
      apples = a;
      cupsSugar = s;
      myCrust = c;
   }

   public double calcWeight(){
      double appleWt = apples * 0.5;
      double sugarWt = cupsSugar * 0.3;
      double crustWt = myCrust.getWt();
      return appleWt + sugarWt + crustWt;
   }
}
```

2. Classes and Objects

1. Making a class
   - Data members
   - Constructor
   - Methods
2. ***Creating objects***
   - ***Memory allocation***
3. Calling methods
   - Passing object arguments
4. Access
5. Static vs. non-static

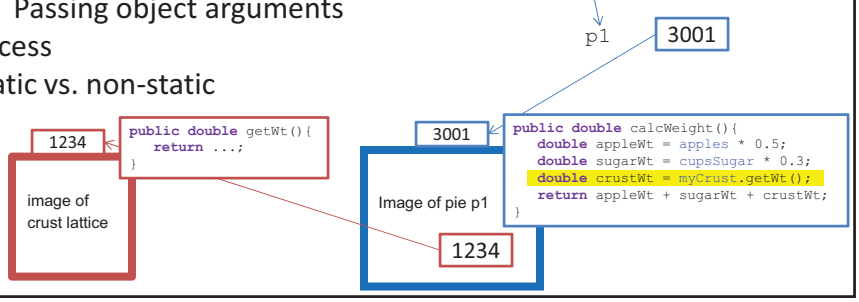
```
public class Test{
  public static void main(String[] args){
    Crust lattice = new Crust(...);
    Crust flat = new Crust(...);

    Pie p1 = new Pie(4, 1.5, lattice);
    Pie p2 = new Pie(8, 3, flat);
  }
}
```



2. Classes and Objects

1. Making a class
   - Data members
   - Constructor
   - Methods
2. Creating objects
   - Memory allocation
3. ***Calling methods***
   - Passing object arguments
4. Access
5. Static vs. non-static

```
public class Test{
  public static void main(String[] args){
    Crust lattice = new Crust(...);
    Crust flat = new Crust(...);

    Pie p1 = new Pie(4, 1.5, lattice);
    Pie p2 = new Pie(8, 3, flat);

    double w1 = p1.calcWeight();
  }
}
```

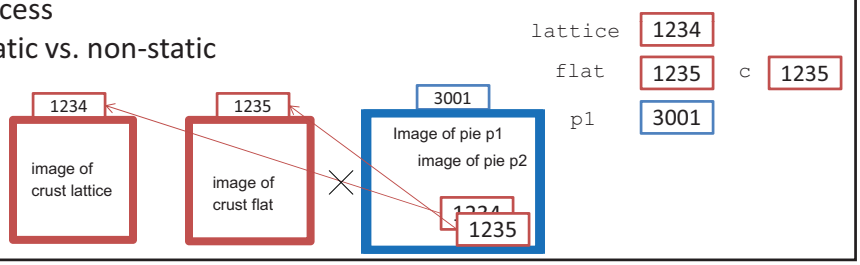


2. Classes and Objects

1. Making a class
   - Data members
   - Constructor
   - Methods
2. Creating objects
   - Memory allocation
3. Calling methods
   - ***Passing object arguments***
4. Access
5. Static vs. non-static

```
public class Test{
  public static void main(String[] args){
    Crust lattice = new Crust(...);
    Crust flat = new Crust(...);

    Pie p1 = new Pie(4, 1.5, lattice);
    p1.setCrust(flat);
  }
}
```

```
public class Pie {
  ...
  public void setCrust(Crust c){
    myCrust = c;
  }
}
```



2. Classes and Objects

1. Making a class
   - Data members
   - Constructor
   - Methods
2. Creating objects
   - Memory allocation
3. Calling methods
   - Passing object arguments
4. ***Access***
5. Static vs. non-static

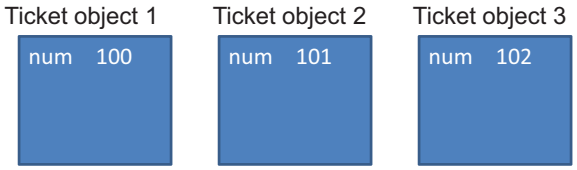
Private: Access only within current class
Public: Access from all classes in project
Package (default): Access from all classes in same package
Protected: Used with inheritance (covered later)

2. Classes and Objects

1. Making a class
   - Data members
   - Constructor
   - Methods
2. Creating objects
   - Memory allocation
3. Calling methods
   - Passing object arguments
4. Access
5. ***Static vs. non-static***

```
public class Ticket {

    private static int total = 0;
    private int num;

    public Ticket() {
        num = 100 + total;
        total++;
    }

    ...
}
```

total 0



3. Arrays and ArrayLists

1. Differences between array and ArrayList
2. Declaration and initialization
3. Assigning / adding an element
4. Accessing an element
5. Retrieving the number of elements
6. Looping over elements

3. Arrays and ArrayLists

1. ***Differences between array and ArrayList***
2. Declaration and initialization
3. Assigning / adding an element
4. Accessing an element
5. Retrieving the number of elements
6. Looping over elements

Array

All about brackets [ ] !

ArrayList

All about methods!

3. Arrays and ArrayLists

1. Differences between array and ArrayList
2. ***Declaration and initialization***
3. ***Assigning / adding an element***
4. ***Accessing an element***
5. ***Retrieving the number of elements***
6. ***Looping over elements***

Array example

```
double[] d = new double[5];
d[0] = 32.0; d[1] = 67.0; ...
System.out.println(d[1]);
int len = d.length;
for(double elem : d)
  System.out.println(elem);
for(int i=0; i<len; i++)
  System.out.println(d[i]);
```

ArrayList example

```
ArrayList<Double> d = new ArrayList<Double>();
d.add(32.0); d.add(67.0); ...
System.out.println(d.get(1));
int len = d.size();
for(double elem : d)
  System.out.println(elem);
for(int i=0; i<len; i++)
  System.out.println(d.get(i));
```

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012