Lecture 6

Variables

Local variables

- a variable that is only relevant for a single method execution can be declared inside the method body:

```
void elections() {
     String answer;
     System.out.println("Who won?");
     answer = Scanner.next();
     if (answer.equals("AZ") {
        System.out.println("pheww");
     }
  }
```

- variable answer is only needed for elections
- answer is not available outside elections
- value is discarded when call to elections terminates (returns)

three kinds of variables

- instance variable:
  - scope is all methods in class
  - automatic initialization with 0 or false (booleans)
- parameter:
  - scope is one method only
  - initialization by call
- local variable:
  - scope is one method or less
  - initialization required

What does the box do?

```
class Printer {
  void hek(int n) {
    String s = "";
    for (int i=0; i<n; i++) {
      s = s+"#";
    }
    System.out.println(s);
  }

  void box() {
    hek(1);
    hek(2);
    hek(3);
    hek(4);
  }

  public static void main(String[] a) {  new Printer().box(); }
}
```

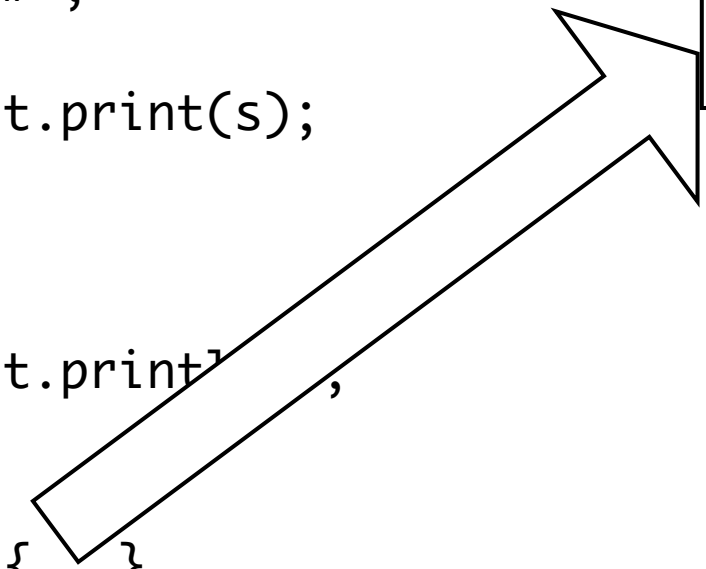
What does the box do?

```
class Printer {
  void hek(int n) {
    String s = "";
    for (int i=0; i<n; i++) {
      s = s+"#";
    }
    System.out.print(s);
  }

  void nl() {
    System.out.println();
  }

  void box() {   }
}
```

```
hek(8); nl();
hek(2); nl();
hek(6); nl();
hek(2); nl();
hek(2); nl();
hek(2); nl();
```



Method types

I. simple

```
void <name>() {
  <body>
}
```

II. parameters

```
void <name>(<type> <var>,..., <type> <var>) {
  <body>
}
```

III. return

```
<type> <name>() {
  <body>
  return <val>;
}
```

IV. combination of II and III

Method examples

I. simple

```
void hello() {
  System.out.println("Hellooooo!");
}
```

II. parameters

```
void repeat(String s, int n) {
  for (int i=0;i<n;i++){ System.out.println(s); }
}
```

III. return

```
double pi() {
  return 3.14159265359;
}
```

IV. combination of II and III

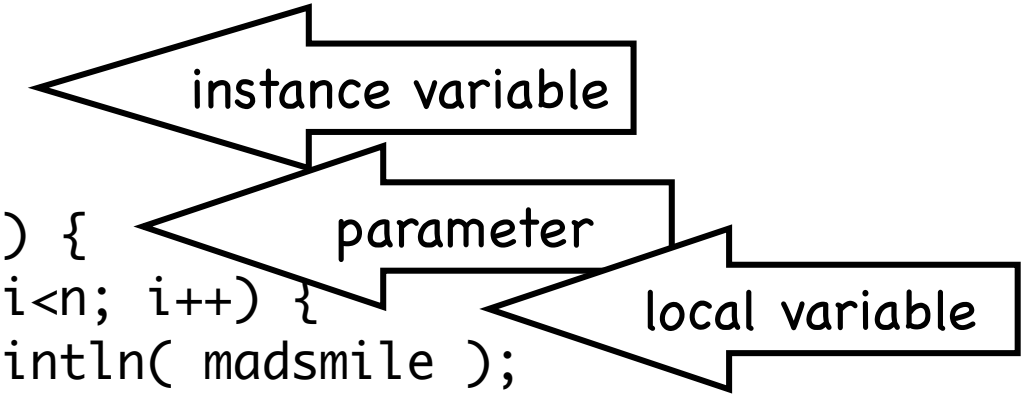
```
int max (int a, int b) {
  if (a>b){return a;}else{return b;}
}
```

Variables

```
class Blah {
   String madsmile;

   void angry( int n ) {
     for (int i = 0; i<n; i++) {
        System.out.println( madsmile );
     }
   }

   void doIt() {
      madsmile = ">:-|";
      angry( 3 );
   }
```



Scope

- scope is the working area of a variable or parameter
- it begins at the declaration and ends at the closing brace
- outside its scope, a variable cannot be referred to
- Example:

```
class Zuip {
  void drunk() {
     if (n > 10) {    // error!
        ...
     ;
  }

  void drink(int n) { ...}
  }
```

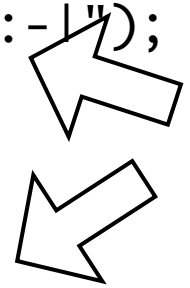
- scope of an instance variable is the whole class
(except the static methods)

Scope 2a

- many variables and parameters with the same name may occur in different parts of your program, without conflict

- Example:

```
void angry(int n) {
  for (int i=0; i<n; i++) {
    System.out.println(">:-|");
  }
}
...
void halve(int n) {
  double i = n/2.0;
  System.out.println("helft is "+i);
}
```



different variables i

Scope 2b

- many variables and parameters with the same name may occur, without conflict

- Example:

```
void angry(int n) {
  for (int i=0; i<n; i++) {
    System.out.println(">:-|");
  }
}
...
void halve(int n) {
  double i = n/2.0;
  System.out.println("helft is "+i);
}
```

different variables n

Scope 3

```
class ScopeExample {
    int n;

    void blah( int p ) {
        System.out.println( p );
        int q = 5;
        n = q * p;
        // etc.
    }

    void rhubarb( ) {
        String n = “twee”;

        System.out.println( n + n );
        // etc.
    }

    void plauder() {
        System.out.println( n*n );
    }
}
```

n

p

q

n

which variables when?

more or less in order of preference

- local var: scratch place
- parameter: to get information into a method
- return: to get information out of a method
- instance var: to exchange info between methods
  - need to put more than one value out of a method

Examples

```
class ExampleVars {

    String name;
    int grade1;
    int grade2;

    double average(int a, int b) {
        int double sum;   // local variable
        sum = a = b;
        return sum / 2;
    }

    void read() {
        name = scanner.next();
        grade1 = scanner.nextInt();
        grade2 = scanner.nextInt();
    }

    void print() {
        System.out.println(name+" has for course 1 a "+grade1);
    }

    void printAvg() {
        System.out.println( name+" has as average "+average(grade1. grade2) );
    }

}

// BTW: order of declarations in class is not relevant
```

Notes

- changes in local variables are lost as soon as method returns
- use instance variables for "permanent" changes
- do not change parameters

More notes

- group code that belongs together in method
- break long methods up (> ±15 lines)
- try to find methods with simple interface:
  - few and clear parameters
  - few and clear uses/changes of instance variables

Specifications

- // @pre: what has to hold before the call
- // @returns: description of return value
- // @post: what holds after the call
- // @modifies: what is changed by the method

Specification example 1

```
//@pre: x >= 0

//@returns: √x

double squareroot( double x ) {

   ...

}
```

Specification example 2

```
//@pre: n >= 0 & input contains at least n ints
//@post: a contains the first n ints from input
//@modifies: a

double readinput( int n ) {
  a = new int[n];
  for (int i=0; i<n; i++) {
    a[i] = scanner.nextInt();
  }
}
```

Notes on functions

- functions are methods with return value
- reserve functions for "mathematical" calculations
- side-effects preferably not in functions:

  change of instance variables

  input or output

- pure functions: functions that do not change or consult the state