Methods

Local and Instance Variables

Local Variables

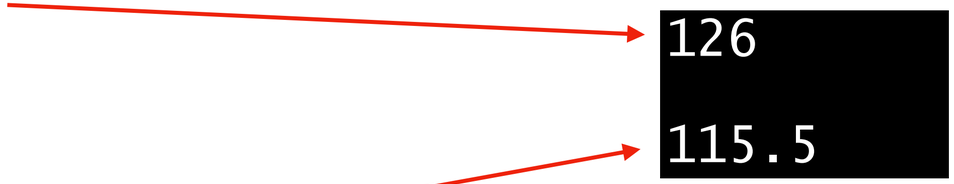
- defined inside the block of code `{ }`, go out of scope
- can have only one optional modifier: `final`
  - once final variable is assigned a value it cannot be changed
- "effectively final": doesn't change value in the scope
  - variable is effectively final if you can put `final` in variable declaration and the code will still compile
- all local variables must be explicitly initialized before used

Instance Variables

- defined on class level, belong to instance of the class
- can have different access modifiers: `private`, `protected`, `public`
- can be marked as `final`, `volatile`, `transient`
- if not initialized, they assume default values depending on type

```
class Item {
  public double tax = 0.2;
  public double getPrice(double inputPrice) {
    double margin = 0.05;
    return inputPrice * (1 + tax) * (1 + margin);
  }
}

public class MyClass {
  public static void main(String[] args) {
    Item item = new Item();
    System.out.println(item.getPrice(100));
    Item specialItem = new Item();
    specialItem.tax = 0.1;
    System.out.println(specialItem.getPrice(100));
  }
}
```



```
// final variables
public double getPrice(double inputPrice) {
   double final margin;
   margin = 0.05;
   return inputPrice * (1 + tax) * (1 + margin);
}
```

ok

```
public double getPrice(double inputPrice) {
   double final margin = 0.05;
   margin = 0.02;
   return inputPrice * (1 + tax) * (1 + margin);
}
```

does not compile

```
// final means that variable reference is constant

// content can be modified


public void printSomething() {
  final int[] a = {1, 3, 5};
  a[1] = 7;
  System.out.println(Arrays.toString(a));
}
```

ok

=> [1, 7, 5]