Programming Syntax
Values

- 3
- 64.6
- 'x' '3'
- "Hello, world!"
- new GP.Attributes.Size(200)
- new GP.Shapes.Rectangle()
Types (and literals)

- `int` 6 32 -5
- `long` 6L 32L -50000000L
- `double` 3.2 6e-12
- `float` 3.2f 6e-12F
- `char` ‘c’ ‘\n’
- `boolean` true false
- `String` “Any string”
Names

• Every name is born with a type:
  • \texttt{int } \texttt{i; GP.Graphics.Oval myOval;}
• Declaration gives a type:
  • \texttt{type-of-thing name-of-thing;}
• Definition gives type and initial value:
  • \texttt{type name = valueExpression;}
• Names refer to values.
• Types determine role of value.
Label Names

- Object names are labels.
- A label may be stuck on one thing — or on nothing (null).
- Two labels may be stuck on the same thing.

myFile

myHouse

myDog

Fido
Scope

• Names can only be known in certain places
  • Your private names, with family, friends, others
  • Fido in your house is different from Fido in CA

• In Java, each level of brackets defines place
  • Each place can have its own names

• Need name to recall value
  • But expression can be used for one time access
Expressions...

- include literals, names
- include result of `new`, methods that return values
- do things with values:
  ```
  myAge + 1  mySize.GetWidth()
  i = 3  new GP.Shapes.Oval()
  ```
- have types and values.
- can be constructed out of other expressions:
  ```
  ((myAge + 1) == 21)
  && (mySize.GetWidth() > 10))
  ```
Statements...

- do not have types or values.
- include declarations and definitions
- include expression followed by ;
- include return; and return expr;
- include method calls that return void (i.e., nothing)
Statements and Expressions

• Actual parameters are expressions
  • ...include names
    $\text{GP.Attributes.Color } c = \text{new GP.Attributes.Color();}$
    $\text{myOval.SetColor(}c\text{);}$
  • ...include creating objects
    $\text{myOval.SetColor(new GP.Attributes.Color());}$
  • ...include methods that return values
    $\text{myOval.SetColor(yourOval.GetColor());}$

• This method call is itself a statement
The Story So Far....

values

expressions

{ 

names

} c

new GP.Attributes.Color()
The Story So Far....

values
{ expressions statements

names

GP.Attributes.Color c = new GP.Attributes.Color();
values

expressions

statements

rules

GP.Attributes.Color c = new GP.Attributes.Color();
myOval.SetColor(c);
public void SetColor (GP.Attributes.Color c)
{
    myOval.SetColor(c);
}

The Story So Far....

values \{
  expressions \rightarrow statements \rightarrow rules
\}

methods

names
The Story So Far....

values

expressions

statements

rules

names

declarations

fields

methods

types

GP.Graphics.Oval myOval;
The Story So Far....

```java
class Cool {
    GP.Graphics.Oval myOval;
    public void SetColor (GP.Attributes.Color c)
    {
        myOval.SetColor(c);
    }
}
```

Robert Duvall and Lynn Andrea Stein
15
Summing up

- Expressions are where the work gets done
- Types determine role of an expression
- Names store the results of expressions
- Statements represent a complete expression
- Procedural methods combine a sequence of statements producing some collective action
- Functional methods can be used to return a result
- Objects provide the context in which methods act
- Parameters provide extra values not otherwise available to methods