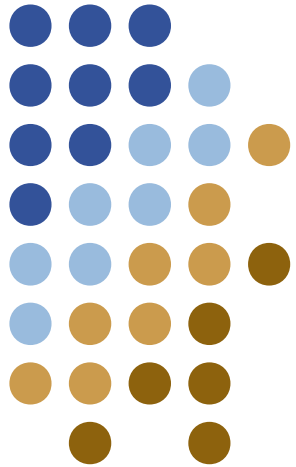
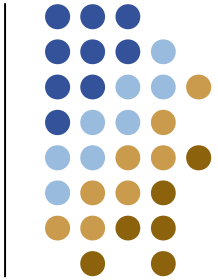
Things I Wrote On The Board

Yannis Smaragdakis, U. Athens
Examples from OO Languages course



Java/C++ References



Java

class A { int i = 0; ... }

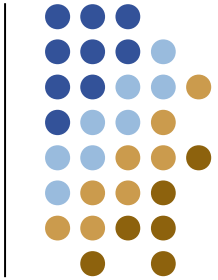
- void foo(int i) { i = 42; }
  int i = 0;
  foo(i); // on return i == 0
- void foo(A a) {a.i = 42; }
  A a = new A();
  foo(a); // on return a.i == 42
- void foo(A a) {
  a = new A();
  a.i = 42;
  }
  A a = new A();
  foo(a); // on return a.i == 0

C++

class A { int i; ... }; // i initially 0

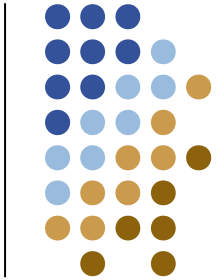
- void foo(int i) { i = 42; }
  int i = 0;
  foo(i); // on return i == 0
- void foo(A &a) {a.i = 42; }
  A a;
  foo(a); // on return a.i == 42
- class A {int i; ... };
  void foo(A *a) {a->i = 42;}
  A a;
  foo(&a); // on return a.i == 42
- class A {int i; ...};
  void foo(A &a) {
  a = A();
  a.i = 42;
  }
  A a;
  foo(&a); // on return a.i == 42

Java Covariant Arrays



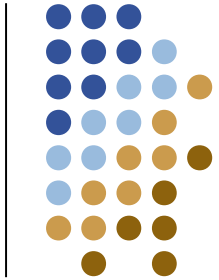
- Dog[] da = new Dog[10];
  Animal[] aa = da;
  aa[0] = new Cat(); // runtime error
  da[0].bark();
- Java: statically type-safe except for casts and covariant arrays
  - a program with no casts, no covariant array use cannot have run-time type error

C++ Overriding, Covariant Return Types



- class A {
  A* foo() {...}
  };
  class B : public A {
  B* foo() {...} // correctly overrides A::foo
  };
- Not a case of overriding:
  class A {
  void foo(Animal& a) {...}
  };
  class B: public A {
  void foo(Dog& b) {...}
  }; // a B cannot do whatever an A can

Named vs. Structural Conformance



- interface Drawable {
  void draw();
  }
  class Cowboy {
  void draw() {...}
  }
  Drawable d = new Cowboy();
  - allowed? Need to say “implements Drawable”?
- Structural conformance can be applied to statically typed languages
  - orthogonal question

Design Pattern: Visitor Example

- class Visitable { void accept(Visitor v) { v.visit(this); } }
- class A extends Visitable { ...
  void accept(Visitor v) { v.visit(this); }
  }
- class B extends Visitable { ...
  void accept(Visitor v) { v.visit(this); }
  }
- interface Visitor {
  void visit(Visitable v);
  void visit(A a);
  void visit(B b);
  }
- class SomeVisitor implements Visitor {
  void visit(Visitable v) {...}
  void visit(A a) {...}
  void visit(B b) {...}
  }

Multithreading

- class A {
  int i;
  synchronized void foo() {... i ...}
  synchronized void bar() {... i ...}
  }
  A a1 = new A();
  A a2 = new A();
  A a3 = a1;
- Can two threads simultaneously execute:
  - a1.foo + a1.bar (no)
  - a1.foo + a2.foo (yes)
  - a1.foo + a2.bar (yes)
  - a1.foo + a3.bar (no)
  - a1.foo + “a1.i = 0” (yes)
  - a1.foo + “synchronized(a3) { a2.bar(); }” (no)
  - a1.foo + “synchronized(a2) { a3.bar(); }” (no)