Homework 4

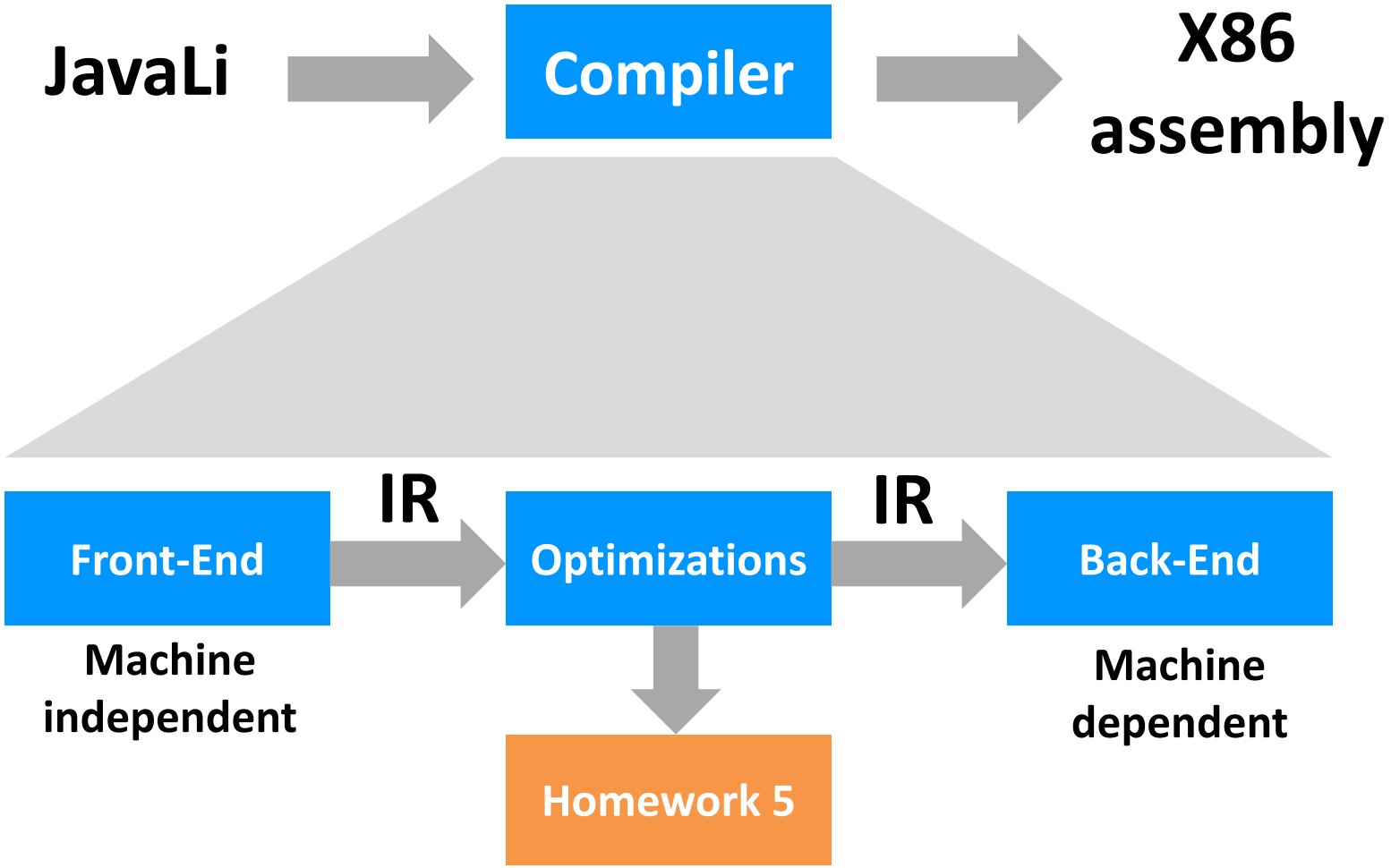
Full code generator

Luca Della Toffola

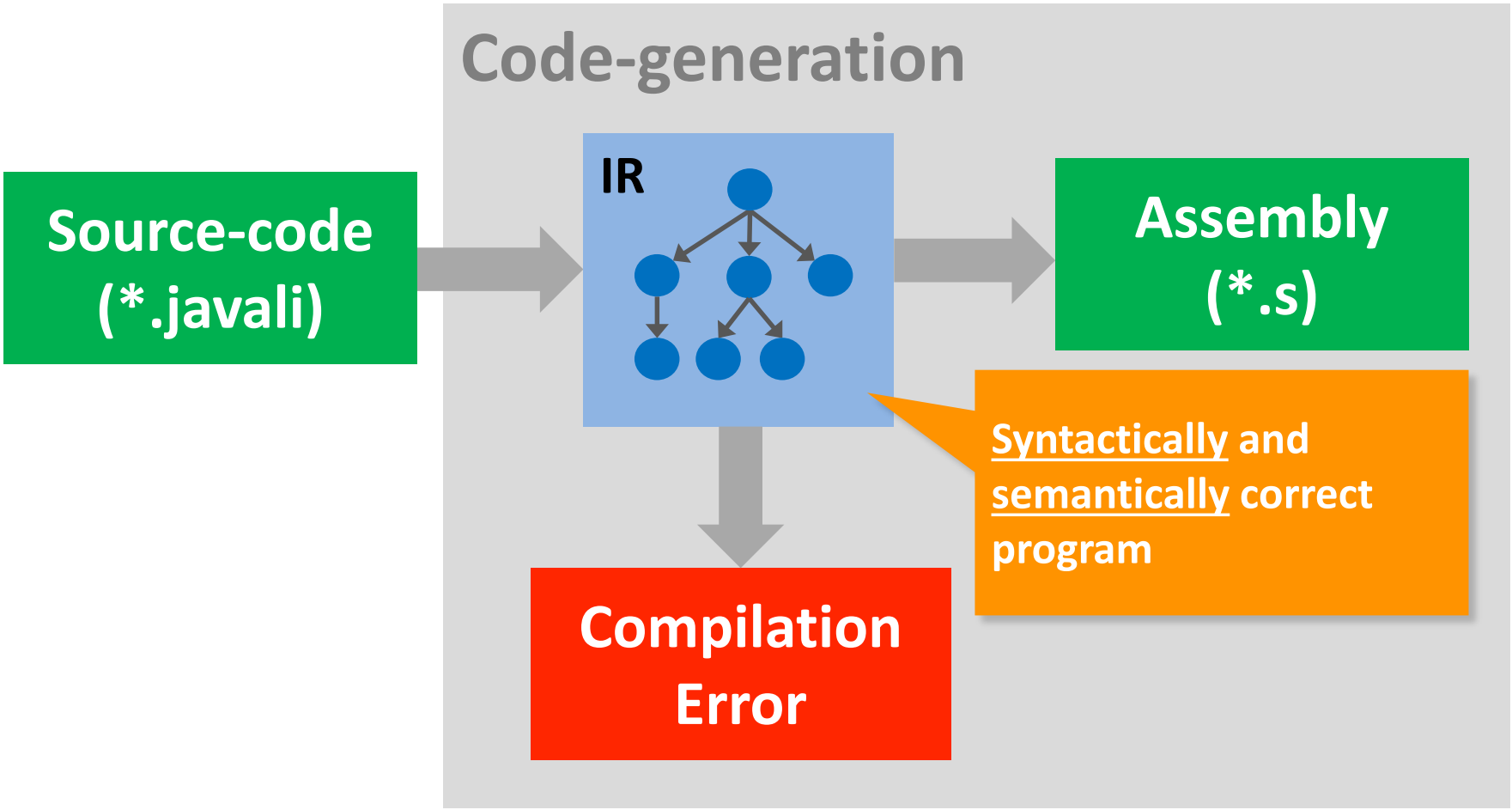
Compiler Design – SS18

Today

Compiler phases



Back-End


Code-generation
Source-code
(*.javali)
IR
Assembly
(*.s)
Syntactically and
semantically correct
program
Compilation
Error

HW4 Tasks

Do not optimize

Constructs to andle

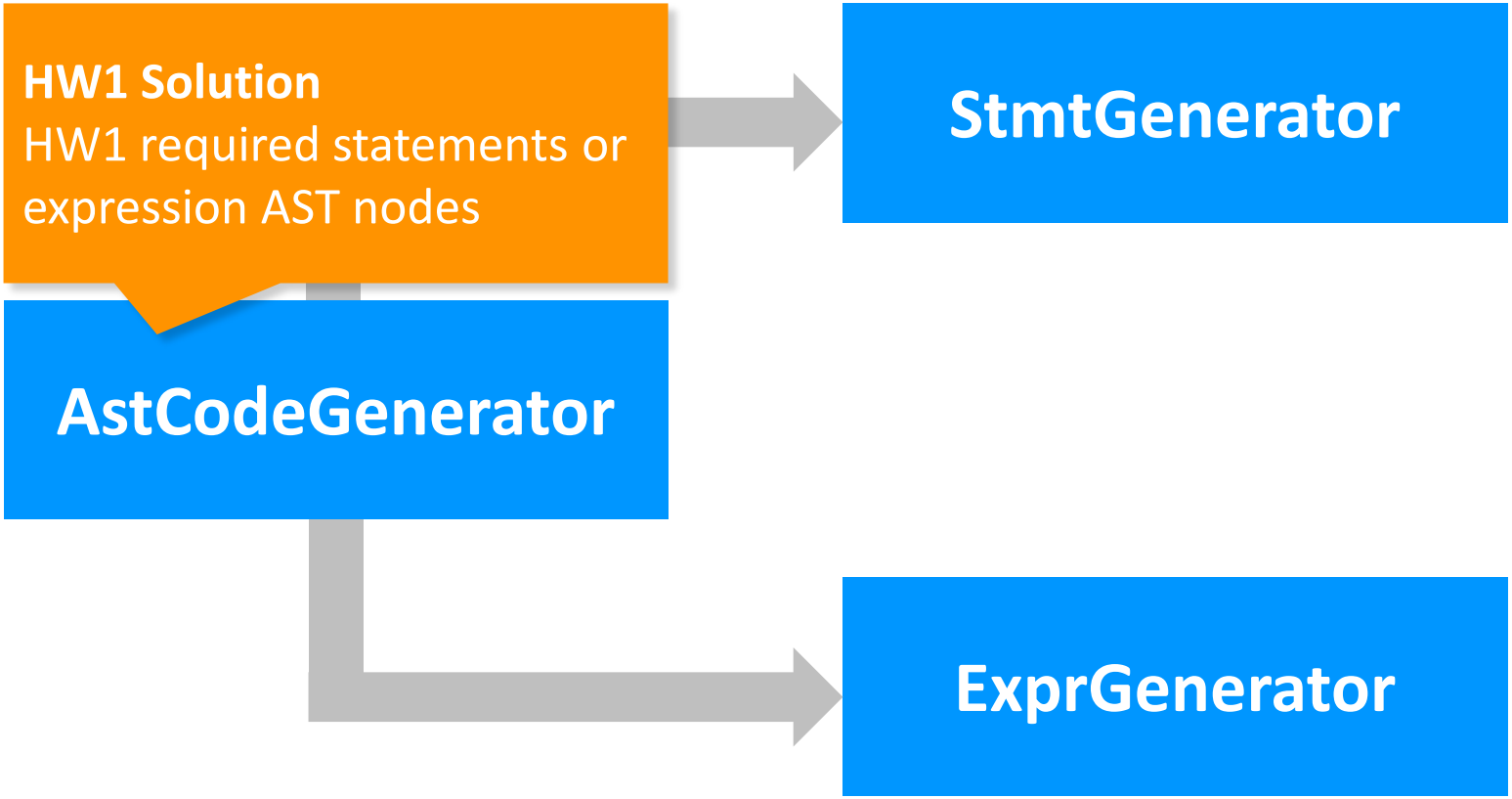
- Object and array creation
- Field reads/writes
- Array indexing
- Method calls (as expressions too)
- Casts
- Control-flow constructs (
- Arithmetic and comparison operators
- Assignment statements

a.b().c();

a[0] > a[2]

ref.a = 5;

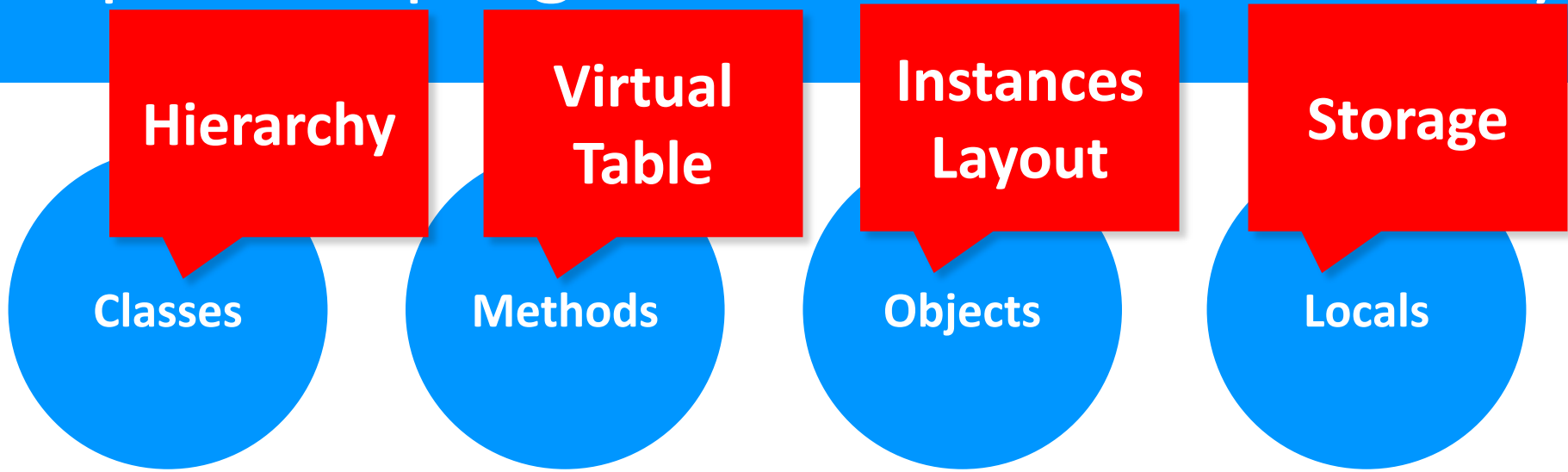
HW4 Fragment



HW4 Tips & Tricks

Additional issues?

Represent program's elements in memory



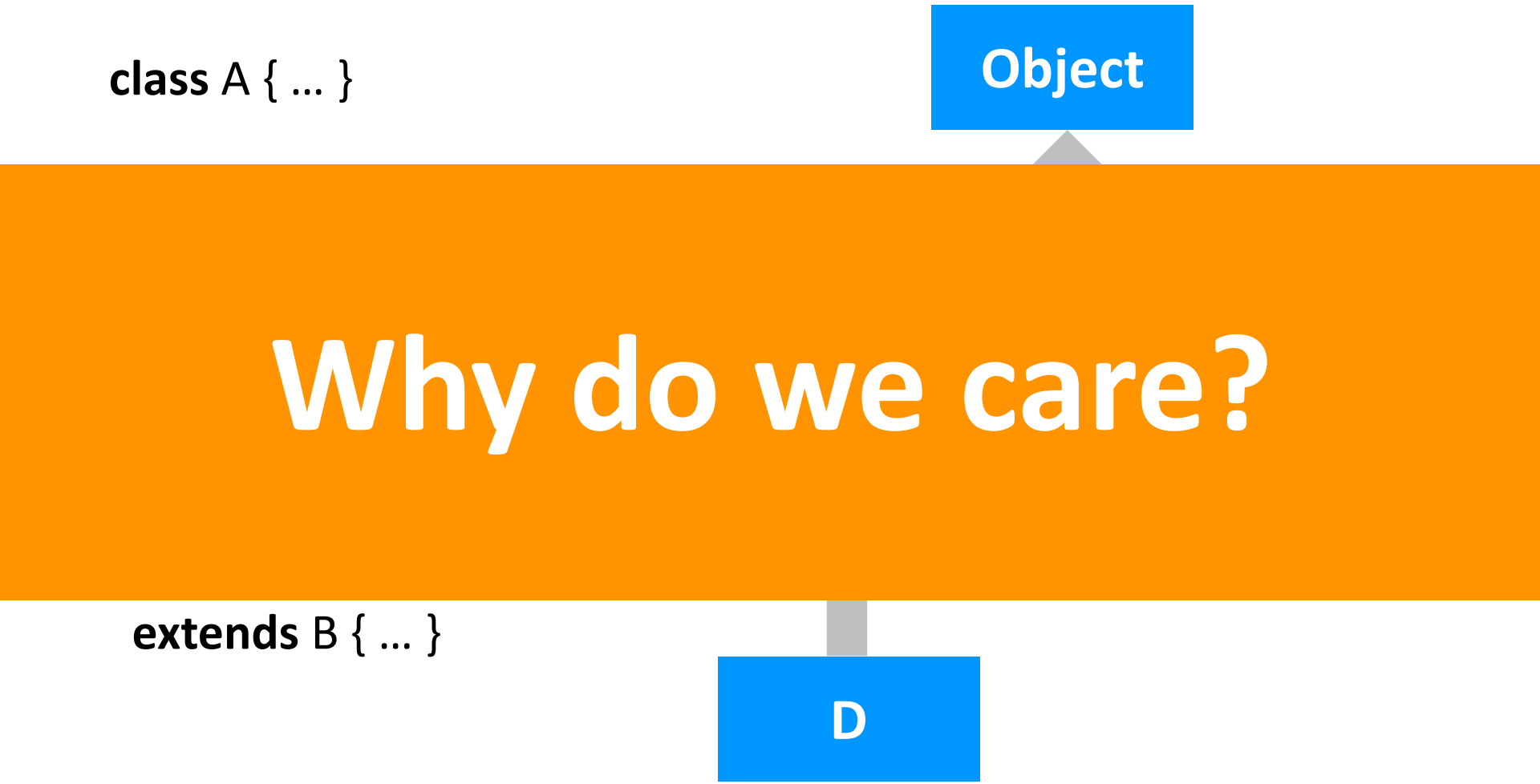
HW4 Tips & Tricks

Additional issues?

Represent program's elements in memory

Classes

class A { … }



extends B { … }

Classes and Casts

```
class A {
  …
}

class B
  extends A {
    …
}
```

```
A a = …;
B b = …;
```

```
a = b; // Upcast – no need for cast
b = (B) a; //Downcast – runtime check need
```

Ensure that **instance** pointed at by **“a”** is a **subtype** of **“B”**

Classes and Casts

```
class A {
  ...
}

class B
  extends A {
    ...
}
```

```
A a = new A();
B b = null;
b = (B) a;
```

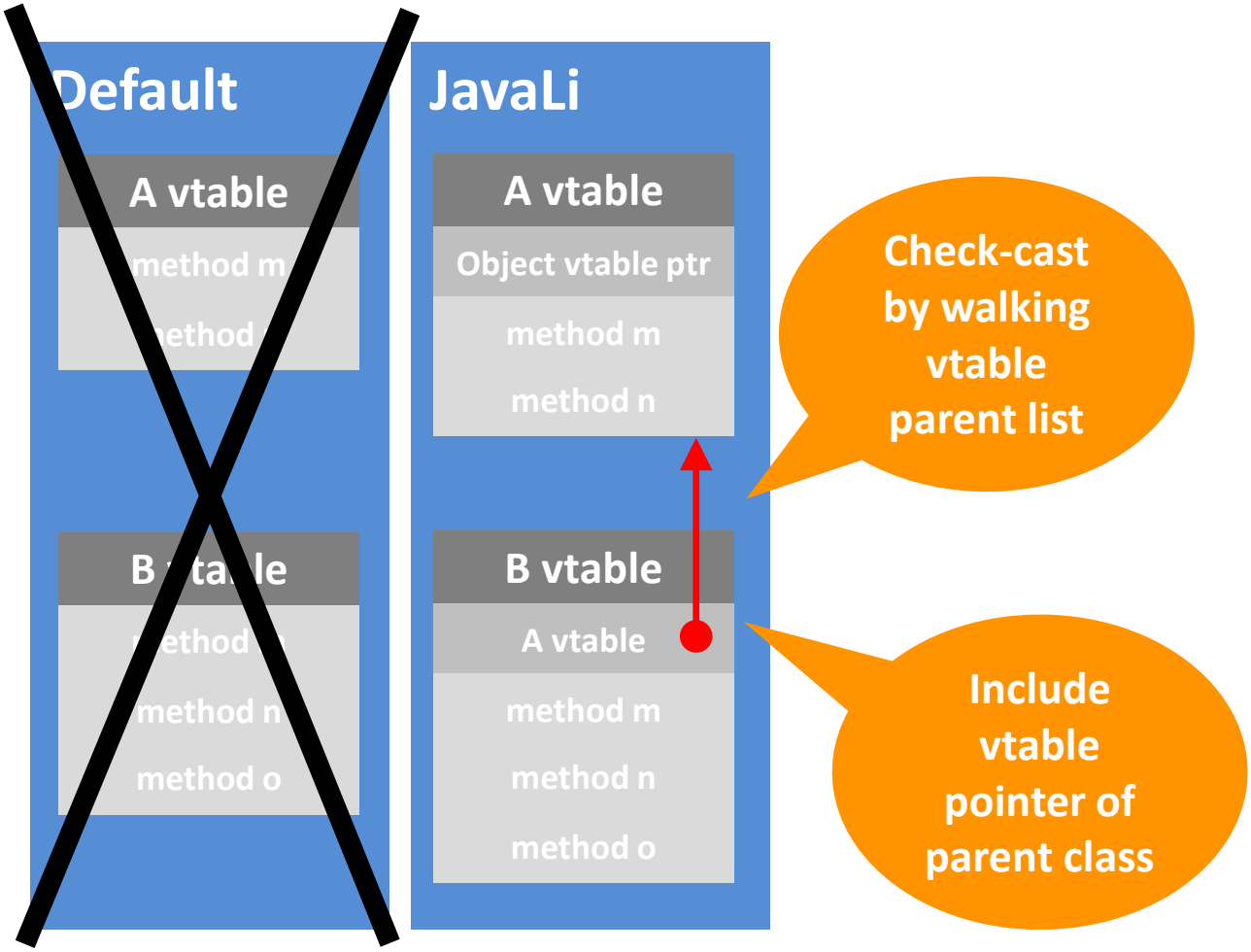
Classes and Casts

```
class A {
  ...
}

class B
  extends A {
    ...
}
```

```
A a = new B();
B b = null;
b = (B) a;
```

Casts and Virtual Tables

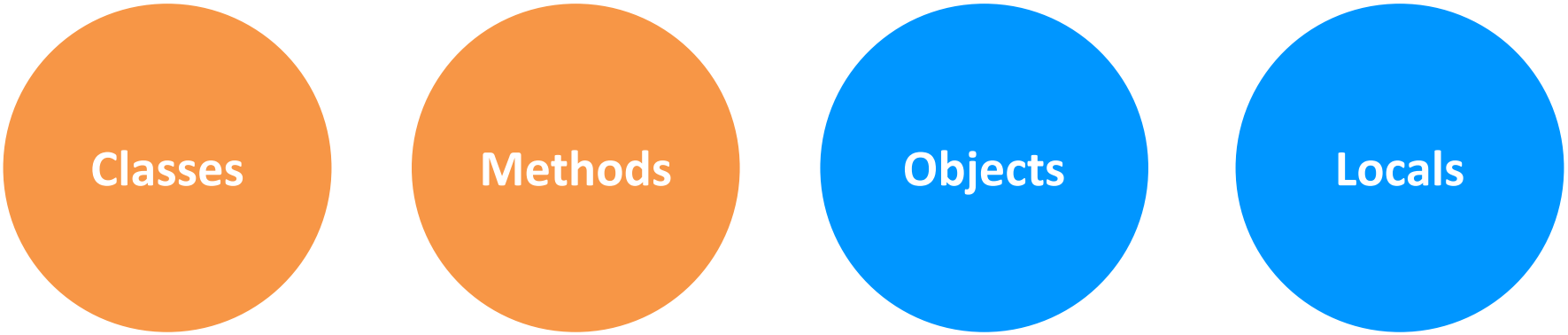
```
class A {
      void m() { … }
      void n()  { … }
}

class B
  extends A {
      void m() { … }
      void o() { … }
}
```



HW4 Tips & Tricks

Additional issues?

Represent program's elements in memory



Methods

```
class A {
  void m() {
    …
  }

  void n() {
    …
  }

}
```



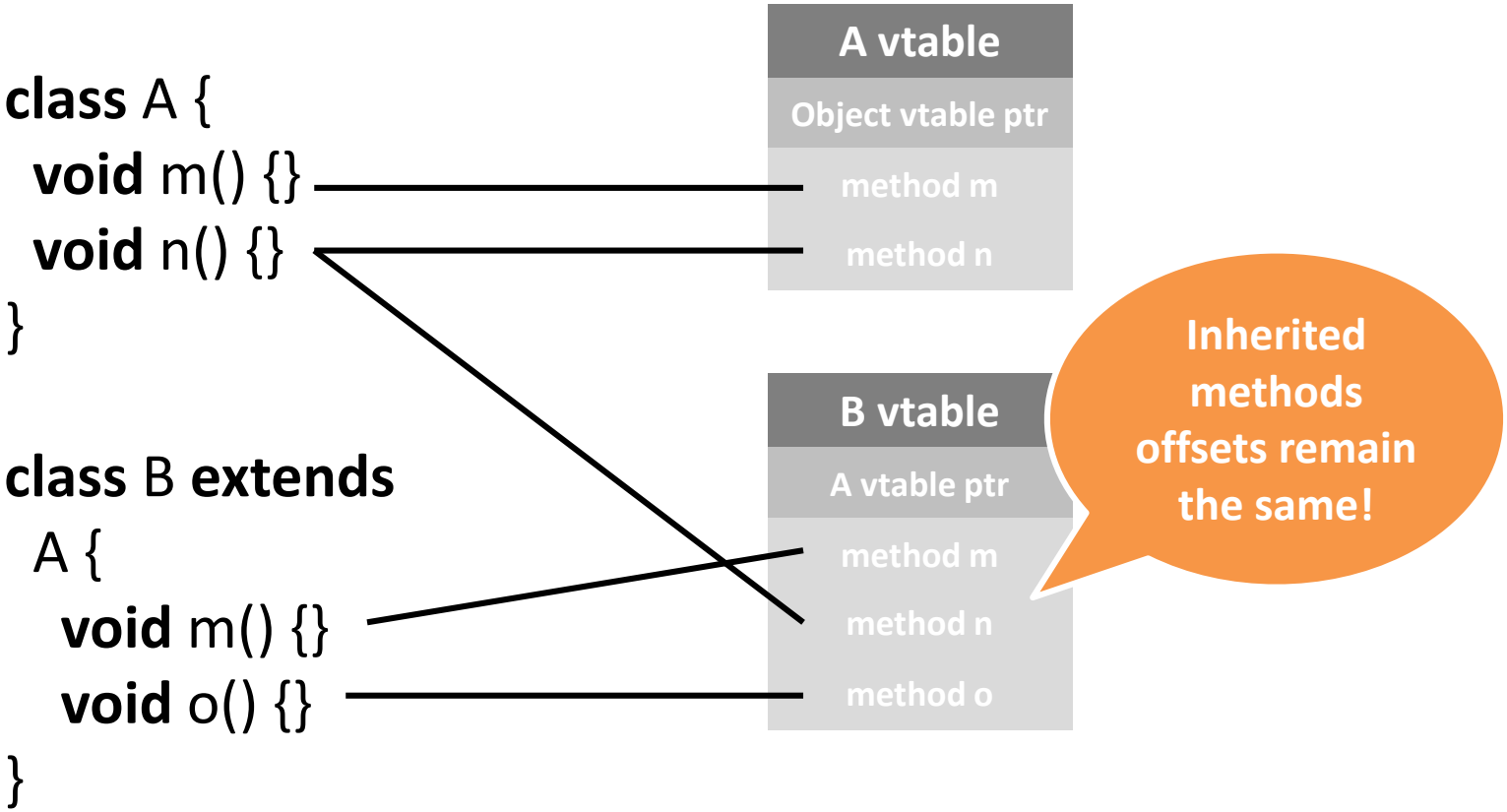
```
.label A::m
load r1
…
ret
```

```
.label A::n
load r2
store r2, r1
…
ret
```

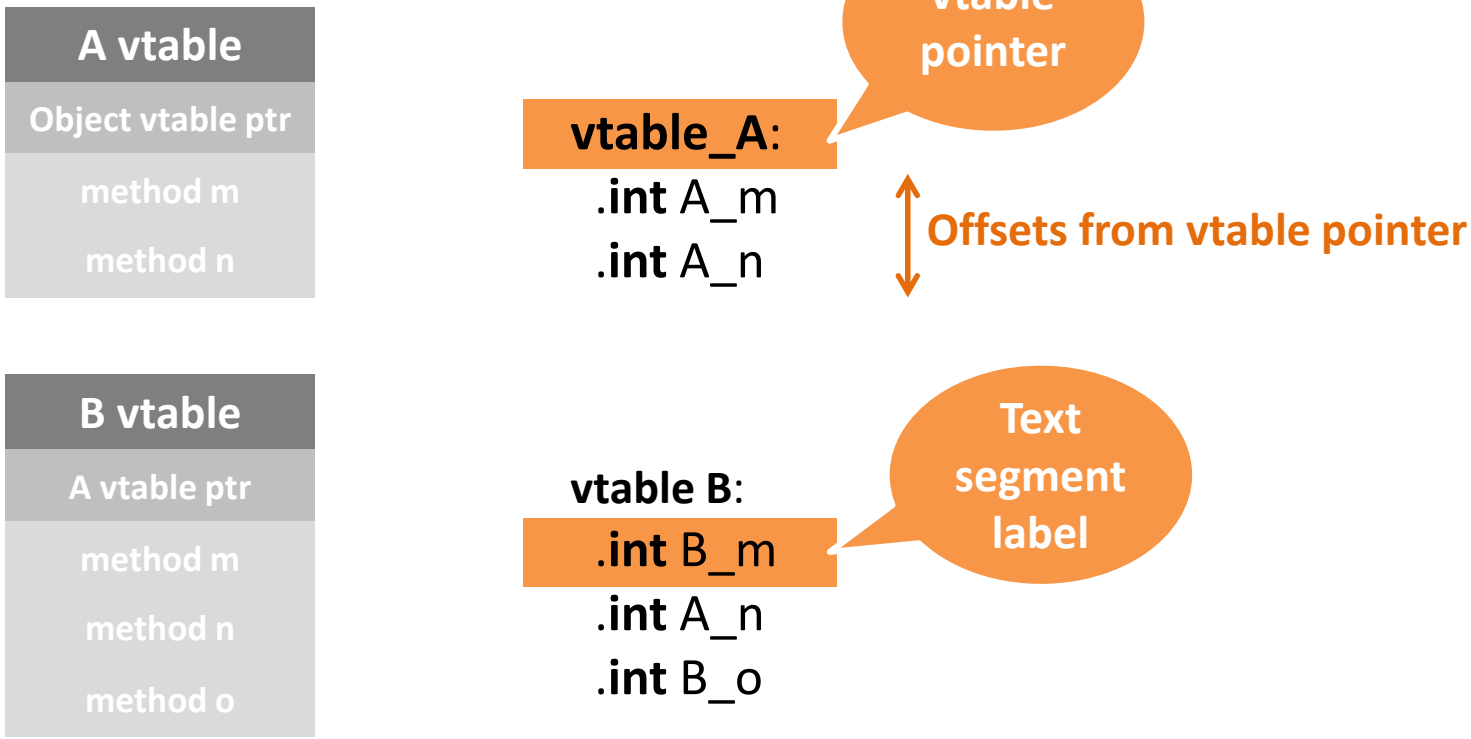
Hint
Exploit the .DATA section again!

Use labels in the vtable

Virtual-Table Layout

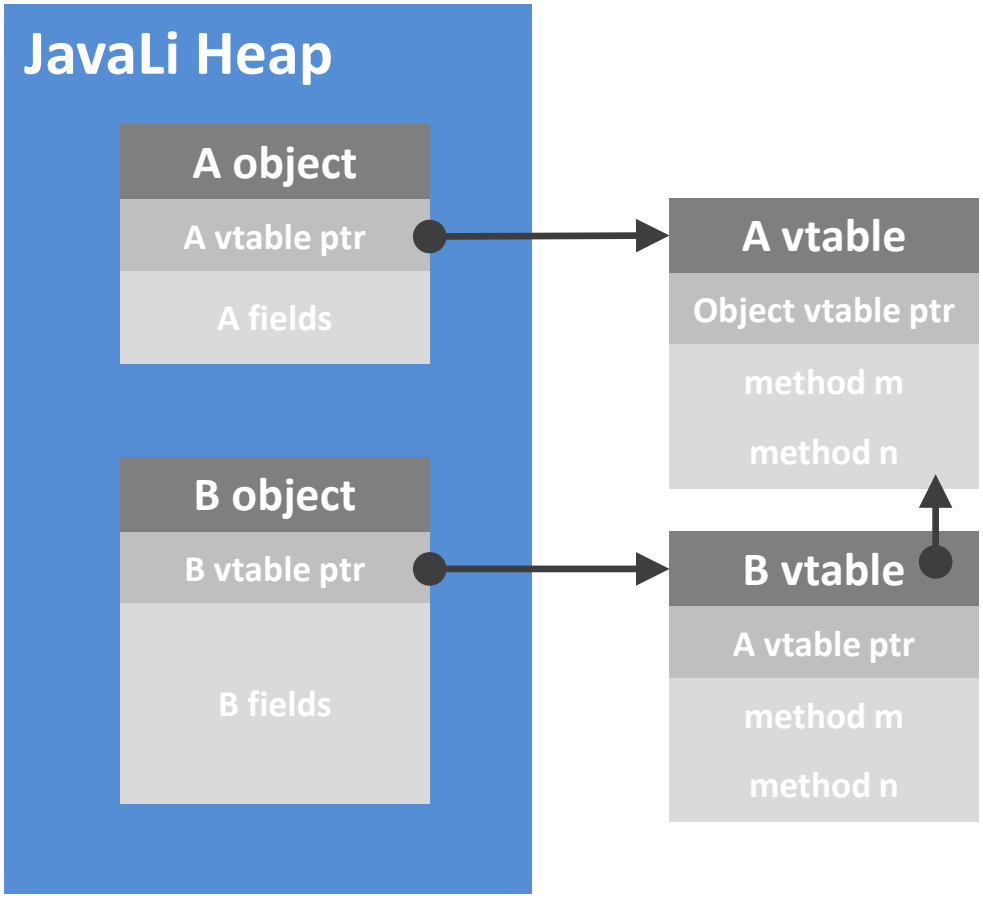


Virtual-Table Declaration



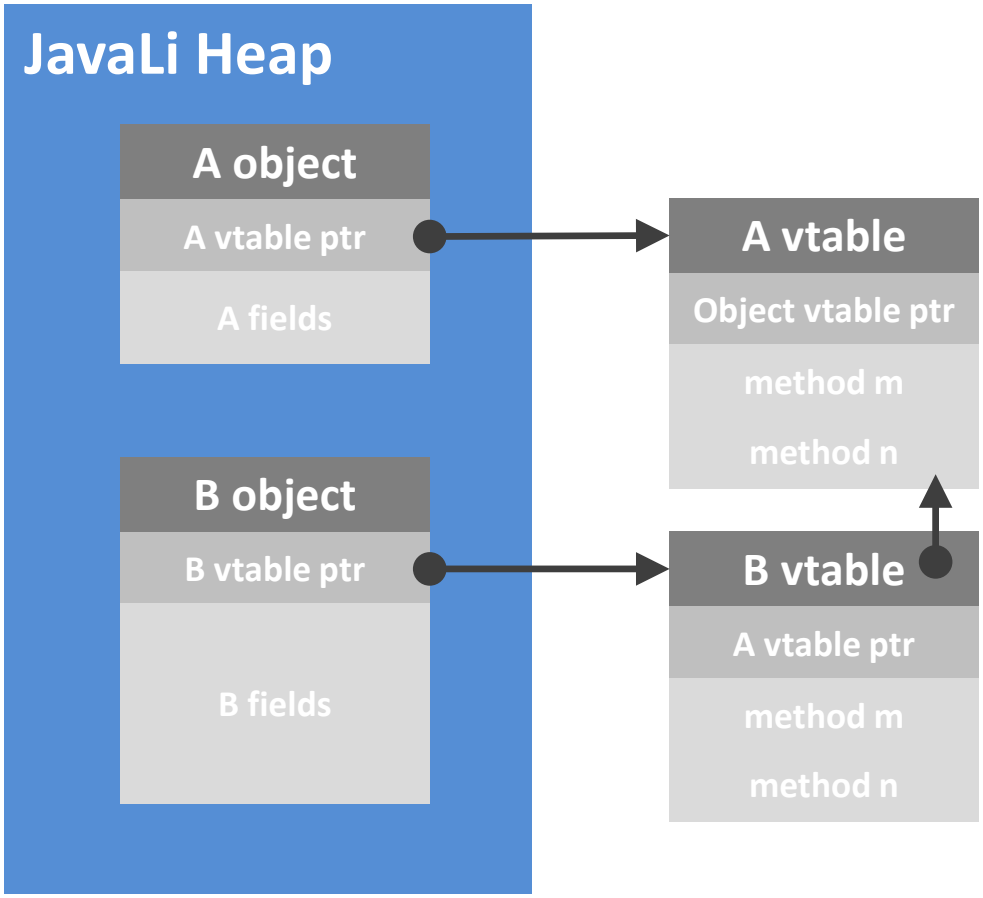
Polymorphic Calls

```
class Main {
      void main() {
         A res;
         res = new ?();
         res.m();
      }
}
```



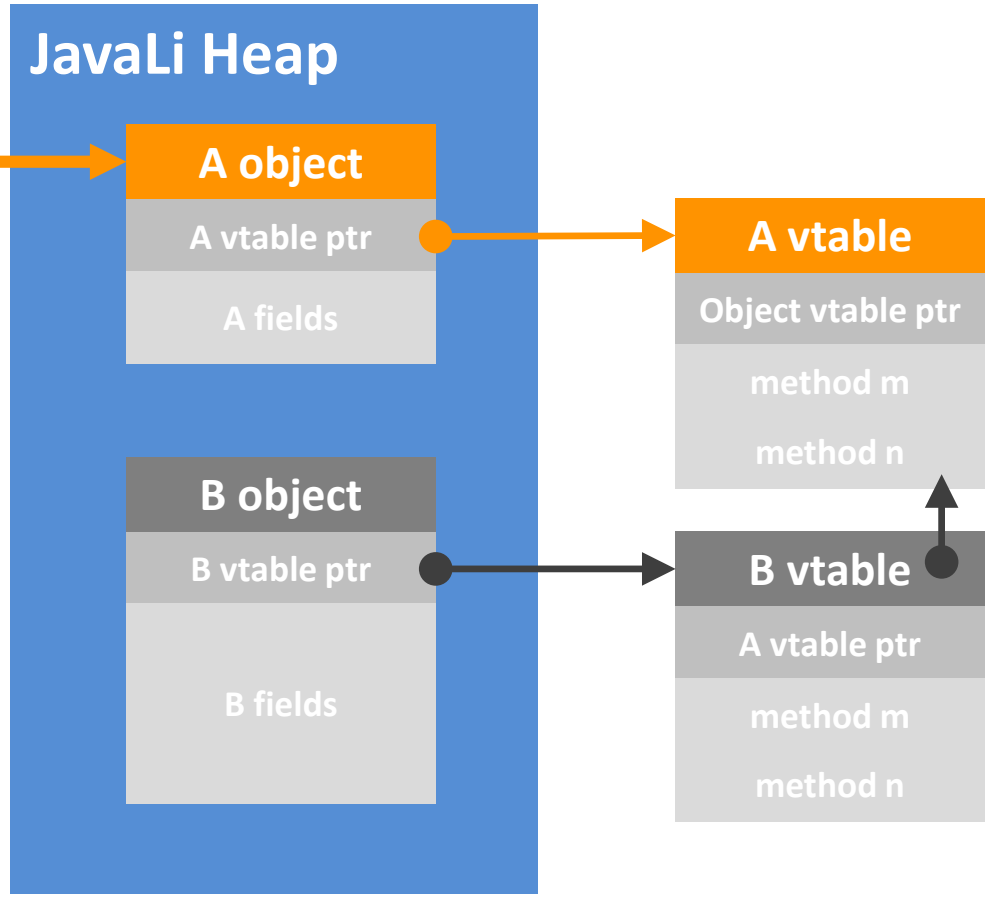
Polymorphic Calls

```
class Main {
      void main() {
         A res;
         res = new A();
         res.m();
      }
}
```



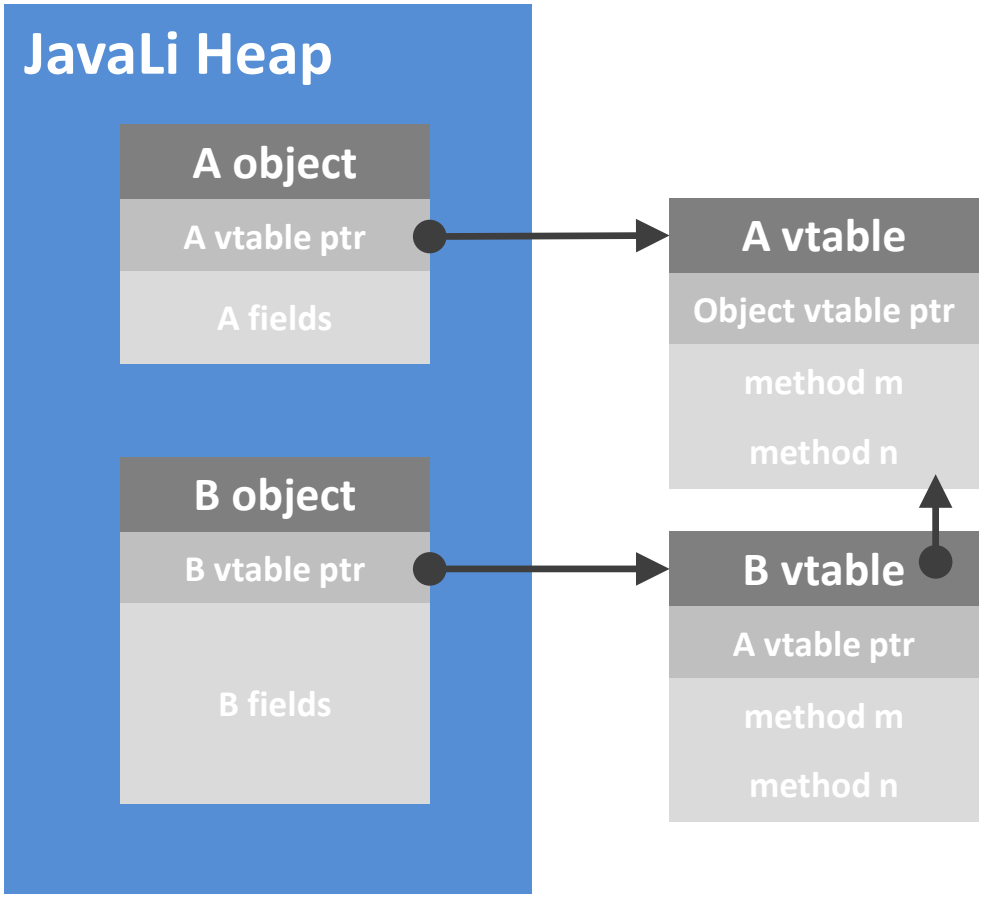
Polymorphic Calls

```
class Main {
    void main() {
        A res;
        res = new A();
        res.m();
    }
}
```



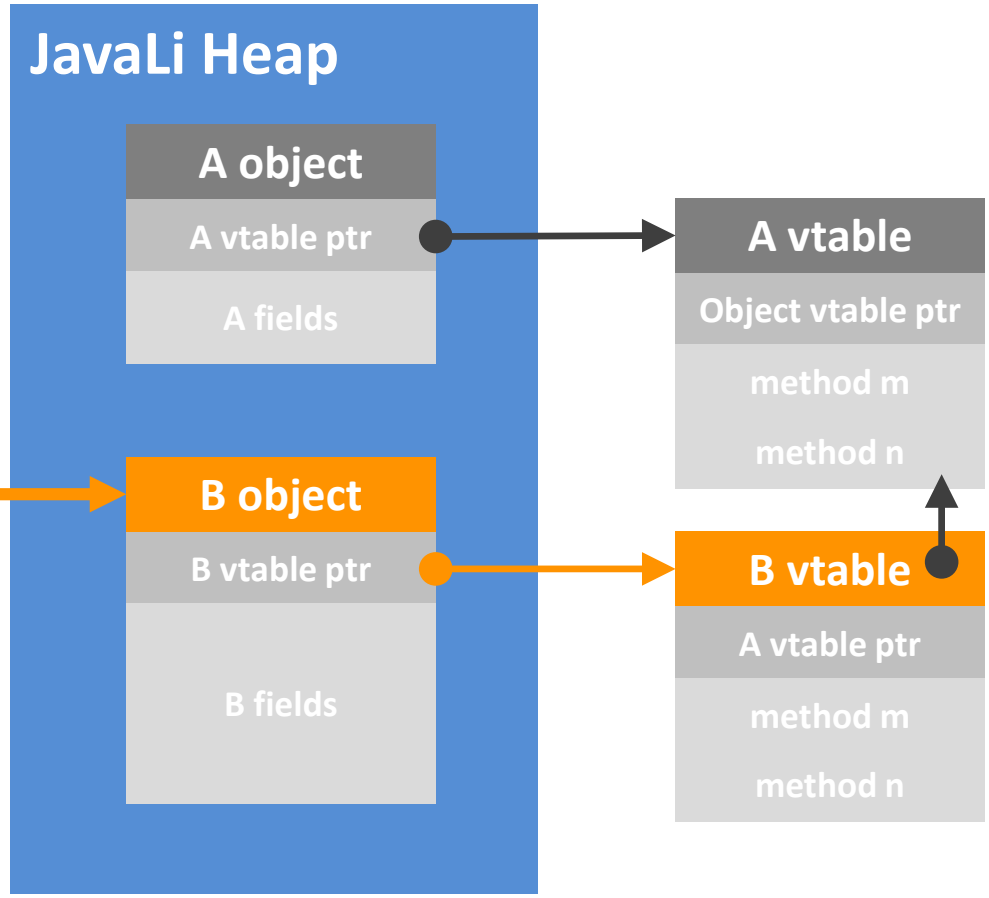
Polymorphic Calls

```
class Main {
      void main() {
         A res;
         res = new B();
         res.m();
      }
}
```



Polymorphic Calls

```
class Main {
    void main() {
        A res;
        res = new B();
        res.m();
    }
}
```



Polymorphic Calls

load instance, r0

load r0[0], r1 *// Get vtable pointer*

load r1[mth_offset], r1 *// Get method pointer*

push arguments *// Prepare the call*

call *r1 *// Execute the call*

load rRes, r1 *// Get the result*

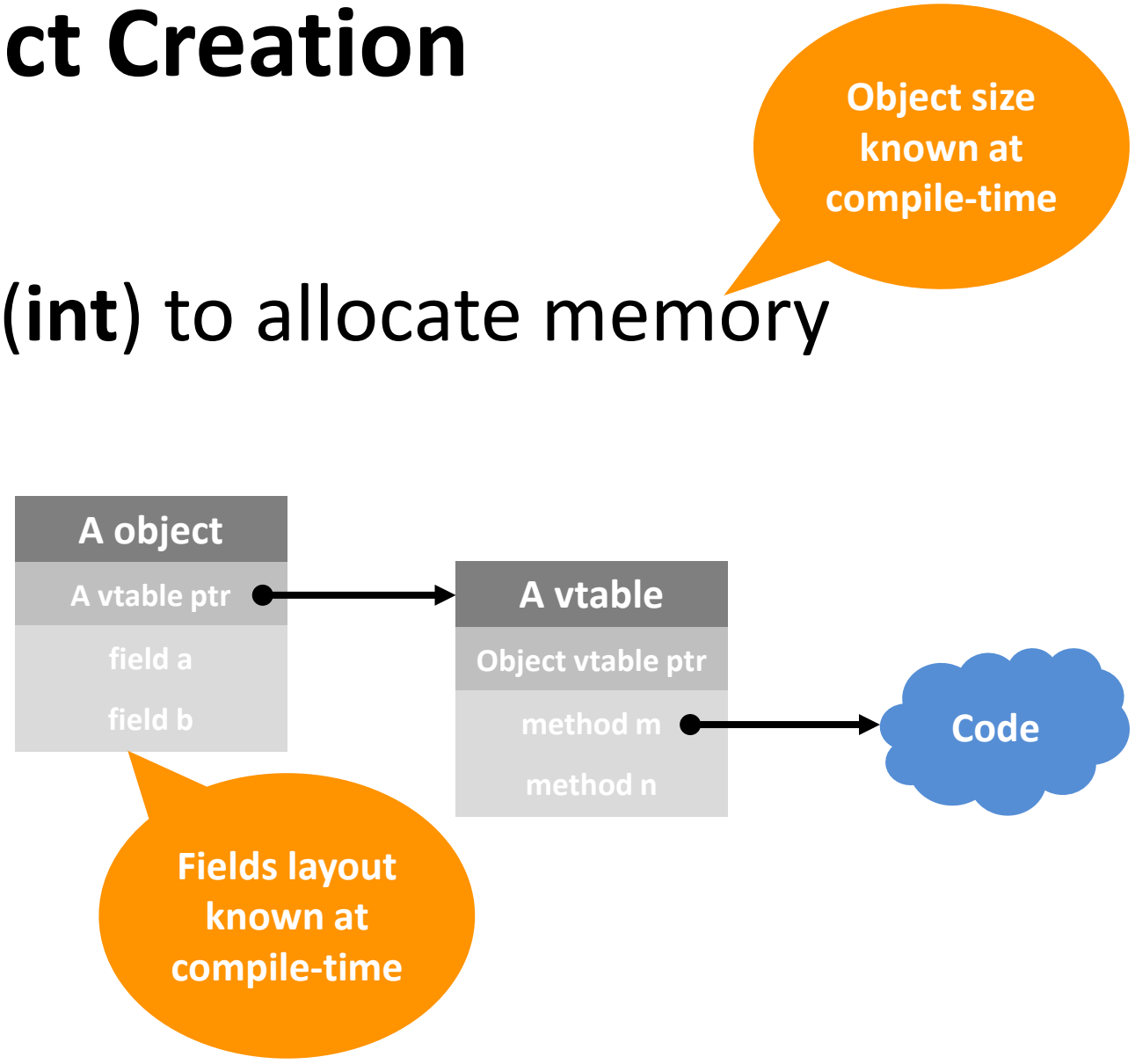
HW4 Tips & Tricks

Additional issues?

Represent program's elements in memory

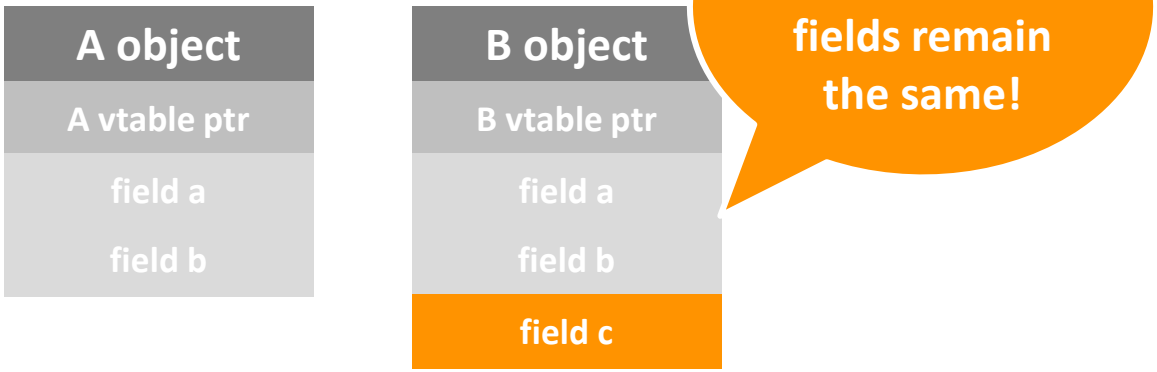
Object Creation

calloc(**int**) to allocate memory



Object Layout

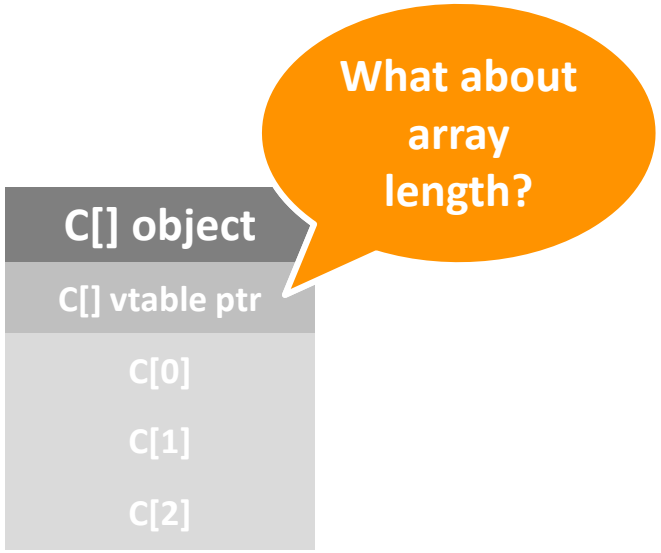
```
class A {
  int a;
  int b;
}

class B extends
  A {
    C c;
}
```



Arrays Layout

They are objects too!

```
class Client {
  C[] array_c;
  …
  array_c = new C[3];
}
```



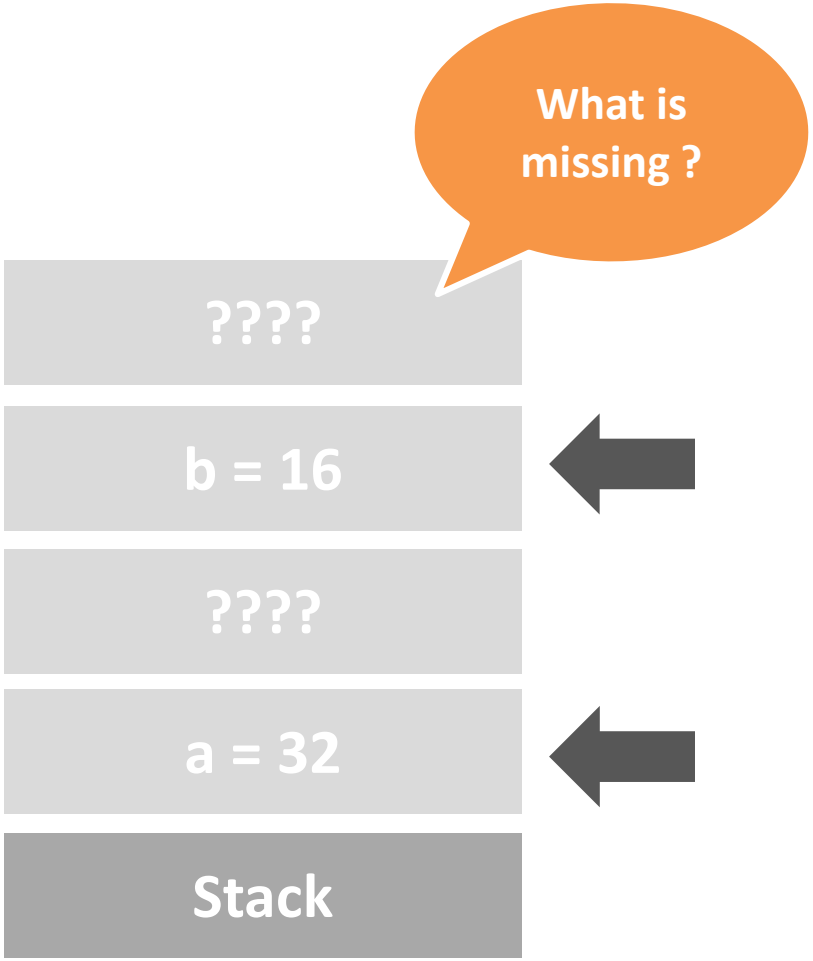
HW4 Tips & Tricks

Additional issues?

Represent program's elements in memory

Local Variables

```
class Main {
      void m(int a) {
            int b;
            b = 16;
            …
      }

      void main() {
            m(32);
            …
      }
}
```



Register spilling

Spilling

Removing value from a register and storing it on the stack

On-the-fly spilling

Done during code generation whenever a register is needed

Explicit spilling

Model the spilling by an assignment to a temporary variable and rewrite the tree

Constructs to Handle

- Object and
- Field r/w
- Array indexing
- Method
- Casts
- Control-flow cons -else)
- Arithmetic and c s
- Assignment statements

Just reviewed

Address + Offset

Address + Offset

Just re bels

Almost done ☺

Almost done 2 ☺