1. When is the cast and field1 access checked in statement (3)?
- **Statically (both)**
- **Statically (cast, field1) & dynamically (cast)**
- Dynamically (both)

```java
class Parent {
    ...
    static int field1;
    static int field3;
}
class Child
    extends Parent {
    ...
    static int field1;
    static int field2;
}
...
1: Parent pobj;
2: pobj = new Child();
3: ((Child)pobj).field1 = 4;
```
2. (we removed *static int field1*; from Child)

Is line 3 an error that will be caught statically?

- **No (not an error)**
  
  • **Note:** If line 2 was: `pobj = new Parent();` then the last answer (yes/dynamically) below would be correct.
  
- Yes, an error caught statically
- Yes an error but caught dynamically

```java
class Parent {
    ...
    static int field1;
    static int field3;
}
class Child
    extends Parent {
    ...
    static int field2;
}
...
1: Parent pobj;
2: pobj = new Child();
3: ((Child)pobj).field1 = 7;
```
3. When is line 3’s cast checked and does it result in an error?
   - statically, no error (Child defines a field1)
   - **statically, error**
   - dynamically, no error (Child defines a field1)
   - dynamically, error

```
javac Parent.java Child.java
Child.java:6: error: cannot find symbol
   ((Parent)cobj).field1 = 7;
  ^
symbol:   variable field1
location: class Parent
1 error
```
• Static dispatch of instance methods (C++, C# for instance methods not marked virtual)
  – The code to the right is Java but imagine that it is C# instead
  – Which implementation of instanceMeth is called for (3) & (4)?

```java
class Parent {
    void instanceMeth() {
        System.err.println("Parent’s IM");
    }
}
class Child extends Parent {
    void instanceMeth() {
        System.err.println("Parent’s IM");
    }
}
public static void main(String[] args){
    Parent obj; //(1)
    obj = new Child(); //(2)
    obj.instanceMeth(); // (3)
    ((Child)obj).instanceMeth(); //(4)
}
```
• **Dynamic dispatch** of instance methods (C++, C# for instance methods marked *virtual*, & all instance methods in Java, Python, Javascript, & others)
  
  – The code to the right is Java but imagine that it is C# instead
  
  – Which implementation of `instanceMeth` is called for (3) & (4)?

```java
class Parent {
    void instanceMeth() {
        System.err.println("Parent’s IM");
    }
}
class Child extends Parent {
    void instanceMeth() {
        System.err.println("Parent’s IM");
    }
    public static void main(String[] args)
    {
        Parent obj; //(1)
        obj = new Child(); // (2)
        obj.instanceMeth(); // (3)
        ((Child)obj).instanceMeth(); // (4)
    }
}
```

Parent's `instanceMethod()` then Parent's instance method
Parent's `instanceMethod()` then Child's instance method
Child's `instanceMethod()` then Parent's instance method
**Child's `instanceMethod()` then Child's instance method**
Review from Last Time: Quiz OO Review

- Solutions: 
  https://sites.cs.ucsb.edu/~ckrintz/classes/s20/cs263/quizsolutions/apr5_oorev_quiz_Sols.pdf

- How dynamic typing differs from static typing (3 ways)
  Possible answers:
  - No type modifier on variable
  - Variable can hold objects/data structures of different types over its lifetime
  - Object attributes must be looked up dynamically on every access
  - Compiler is unable to check and catch many errors statically
  - Type safety only ensured by checking each access dynamically (at runtime)
  - Others?

- Which languages are statically typed and use static type inference? Java, Lua, *ML, *Scala, node.js, PHP, *Go, bash

- obj.meth(4); //translates to meth(obj, 4)
- Rest are dynamic/static dispatch (w/ casting) like the polls... questions?