Ambiguous Arity

```
public class Arity {
  public static void main(String[] args) {
    Example.arity(3, 7); //Error: java: reference to arity is ambiguous
  }
}

class Example {
  public static void arity(Integer n1, int n2) {...}
  public static void arity(int n1, Integer n2) {...}
}
```

Primitive

```
public class Primitive {
  // Okay!
  B[] arrB = {new B()};
  A[] arrA = arrB;

  int[] arrInt = {1,2,3,4};
  double[] arrDbl = arrInt; // Compile time: not ok; int[] is not
                            // a subtype of double[]

  /**
   * Interestingly this makes sense because int is not a subtype of double.
   * Java primitives have no class, and primitive types are not objects.
   * Java's primitive conversions are allowed/disallowed based on whether
   * information is lost.
   *
   * In this case, int[] is an object, so is double[] - but they are not subtypes.
   */
}

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

Static Generic

```
/**
 * You can't use a class's generic type parameters in static methods or
 * static fields.
 * The class's type parameters are only in scope for instance methods and
 * instance fields
 */
class StaticGeneric<T> {
  static int x = 1;
  static T y; // Compile error: non-static type variable T cannot be referenced
```

```
              // from a static context.
  static T foo(T t) {return t;}; // Compile error: non-static type variable T
                                 // cannot be referenced from a static context.
}
```

Interface Casting

```
public class InterfaceCasting {
  public static void main(String[] arg) {
    // Declare interfaces
    Root<?> origin = new T();
    /**
     *  if there exists a supertype X of T, and a supertype Y of S,
     *  such that both X and Y are probably distinct parameterized types,
     *  and that the erasures of X and Y are the same, a compile-time error occurs.
     *  TlDr: Java knows for sure that there cannot be a sub-type S that satisfies
     *        two or more implementation of the same interface.
     */
    X xx = (X) new S(); // Compile time: Incompatible types
    Z zz = (Z) new S(); // Compile time: ok Z is not parameterised.

    /**
     * You can cast any non final class to a non parameterised interface
     */
    Z zOk1 = (Z) new T();
    Z zOk2 = (Z) new S();

    X xOk1 = (X) new V(); // X is parameterised. Compile time: ok because V's
                          // supertype Z is not parameterized
                          // See class U
    Z zNotOk = (Z) new Integer(2); // Compile time: not ok because Integer is a
                                   // final class that doesn't inherit Z.
                                          // i.e. cannot be extended.
  }
}

interface Root<E> {}
//PARAMETERISED
interface X extends Root<Number> {}
class T implements X{}
//PARAMETERISED
interface Y extends Root<String> {}
class S implements Y{}
//NON-PARAMETERISED
interface Z<E> extends Root<E> {}
class V<E> implements Z<E>{}
class U extends V<Number> implements X {}
```