CS 2110: Parametric Polymorphism / Generics

- Prelim almost graded.
- Trouble spot: dynamic type

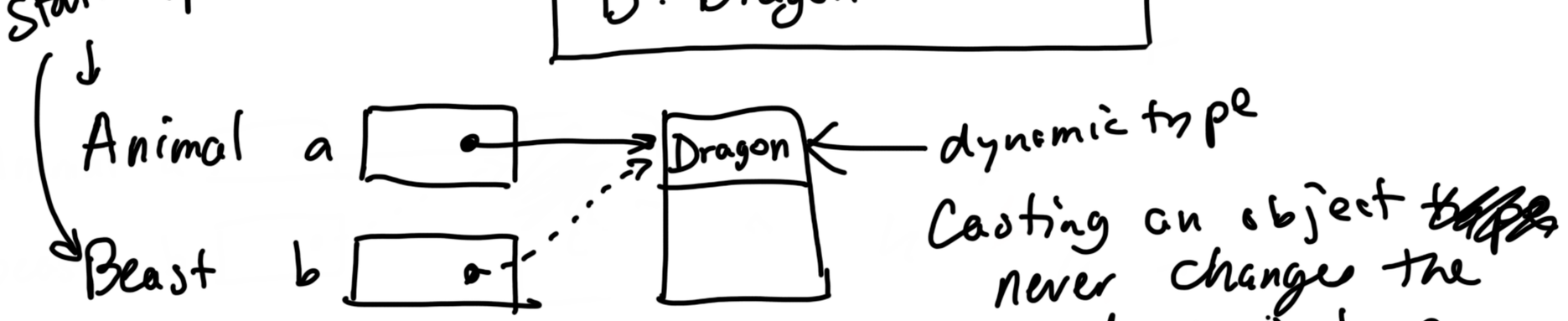
```
Animal a = new Dragon();
Beast  b = (Beast) a;
```

Animal
↑
Beast
↑
Dragon

Dynamic type of object referenced by b?

A: Animal
B: Beast
C: something else
D: Dragon

Static type



Casting an object never change the dynamic type

Driver: Collection Framework

Java 1.4 Collection

```
interface Collection {
    boolean contains(Object o);
    boolean add(Object o);
    boolean remove(Object o);
    Iterator iterator(); //support enhanced for loop
}
Collection c = new ArrayList();
c.add(3); // mte: auto boxed into Integer
c.add("hi");
for (Object o : c) {
    Integer i = (Integer) o; // can fail!
}
```

Generics / Parametric Polymorphism

OO: subtype polymorphism

Client: sees 1 type
Impls: ≥ type (= polymorphism)

Parametric polymorphism (generics)

Impl: 1 type (type parameter)
Clients: use with ≥ 1 type

↳ Not classic OO, added late

Arrays: better

```
Integer[] c = new Integer[2];
c[0] = 3;
c[1] = "hi";   //static error.
for (Integer i : c) {
    ⋮
    use i
    ⋮
}
```

Arrays: a built-in parameterized type

Integer[] ≈ Array<Integer>

type argument

User-defined parameterized types? Collection<Integer>
Collection<String>
Collection<Collection<String>>

Limitations

In a generic context (with parameter type T)

1. No primitive args

   ~~Collection<int>~~

   workaround: use boxed equivalent type

   Collection<Integer>

   int ⟷ Integer (auto boxing →, unboxing ←)

2. Can't create a T

   ~~new T(..)~~

3. or T[]

   new T[n]; ✗

   workaround:

   T[] a = (T[])new Object[n];

   or C<T>[] a = new Collection<T>[n]; (unsafe) ✗

   = new Collection[n]; (raw type (unsafe))

4. Can't cast using T

c instanceof T ✗

c instanceof C<T> ✗

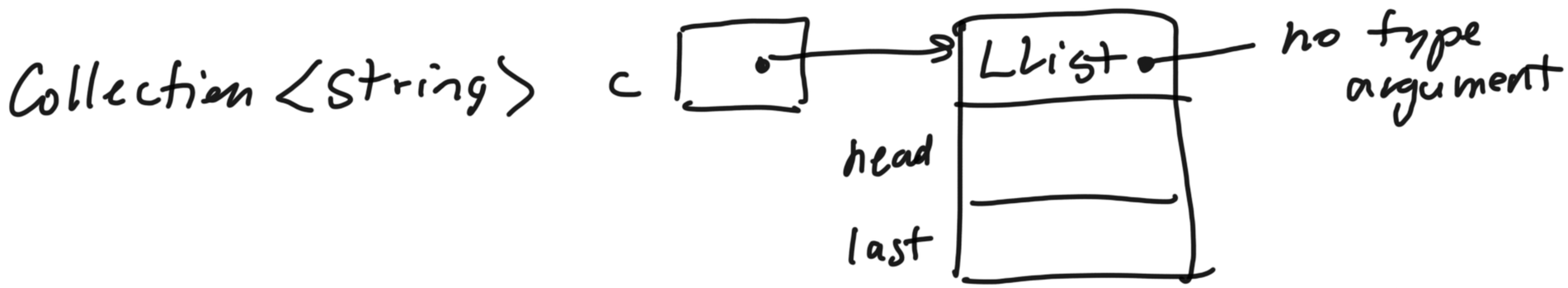
c instanceof C

(unsafe: raw type)

Erasure

Java has **erased generics**

```
Collection<String> c = new LList<String>();
```




```
Collection<Integer> i = (Collection<Integer>)(Collection) c;
int x = i.first(); // ok! runtime error
```

unsafe casts

⇒ object doesn't "know" it holds strings

vs. **reified generics** (eg. C#) type arguments are "real" at run time.