

COMP 202

Generics

CONTENTS:

- Objects and casting
- Generics
- ArrayList again

Objects and Casting

- All instances of all Classes in Java are “also” of the class *Object*
 - we won't get into the full hierarchy
- For a generic array class, useful for any type, use an array of Objects
- But if you insert a String, how do you get back a String?
 - `ArrayList aList = new ArrayList();`
`aList.add("foo");`
`String s = aList.get(0); // error`
 - need to *cast* the result value back to its original Class
 - `String s = (String)aList.get(0); // proper cast used`

Objects and Casting

- If we insert the wrong object type or cast to the wrong type we get an error:
 - `ArrayList aList = new ArrayList();`
`aList.add("foo");`
`String s = (Cat)aList.get(0); // exception`
 - The above results in *ClassCastException* being thrown
- Casting problems are common runtime errors
- Java now warns about using *unchecked* operations
 - ie using casts when there's an alternative that the compiler could check for you...

Generics

- A better solution statically specify the type the elements really are, even though the collection is generic:
 - generic types, also known as parameterized types
 - or in C++, templates
 - idea is to make the element type a “parameter” of the collection type
 - uses a special syntax

Generics

- Element type goes in angle-brackets with the collection type
- For example:
 - `ArrayList<String> listOfStrings;`
`ArrayList<Cat> litter;`
 - here an `ArrayList` of only `String` objects and only `Cat` objects
- The element type must also be specified in the new expression:
 - `listOfStrings = new ArrayList<String>();`
 - `litter = new ArrayList<Cat>();`
- Now the compiler knows `listOfStrings` only accepts `String` objects, and `litter` only accepts `Cat` objects.

5

Generics

- Do not need to cast; ensures type safety at compile-time
 - easier to find bugs than from a runtime exception
- For example:
 - `ArrayList<String> aList = new ArrayList<String>();`
`aList.add("foo");`
`String s = aList.get(0); // no cast, no error!`
- If you try to add non-String objects you then get a compile-time error:
 - `ArrayList<String> aList = new ArrayList<String>();`
`aList.add(new Cat()); // won't compile`

6

The ArrayList<E> Class

- `ArrayList<E>()`
- `boolean add(E obj)`
- `void add(int index, E obj)`
- `E remove(int index)`
- `E set(int index, E obj)`
- `void clear()`
- `boolean contains(Object obj)`
- `int indexOf(Object obj)`
- `E get(int index)`
- `boolean isEmpty()`
- `int size()`

7