Lab 6: Generics and Sorting

Friday, October 9, 2020 11:59 AM

Generics

→ placeholder to use any object data type w/in a class
  * can't use primitives (int, char, float, etc)

→ declare parameter in class heading
  Public class ArrayList<E> ...

→ must provide type when instantiating
  ArrayList<String> list;
  → default to object
  → "raw types" warning — not allowed in 351

→ at compile time, parameter replaced w/ type
  * no more references to parameter
  LinkedCollection <Particle> col;
  → easier to debug — all errors/warnings at compile time

Restrictions

→ cannot create arrays of type E
  i.e. E[] a = new E[size]
  instead:
  E[] a = (E[]) new Object[size];
  ↳ object type
  → comes from Object

→ will always give unchecked cast warning ⚠
  → suppress warnings at top of method
  "unchecked"

→ Student extends person
  "ClassCastException" → cast not allowed *

→ cannot create generic objects
  i.e. E var = new E();

→ cannot check if object is generic type
  i.e. obj instanceof E

→ Wild card: unknown type
  <? extends E>
  Students extend person
  Professor extend person
  TA extends Student extend person
  E = Person

... to be able to

E = Person

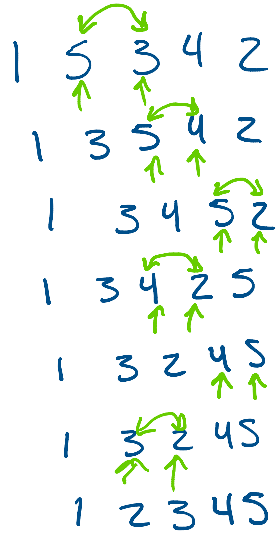
Particles → would not be able to add

ArrayList list = new ArrayList();
→ raw type: did not give type to replace parameter
→ will work: replace all parameters w/ "Object"
→ not allowed in 351

Sorting

→ Bubble Sort
→ while swaps have been made:
→ loop through all elements:
→ compare element 2 one after
→ if out of order, swap

Swap: F



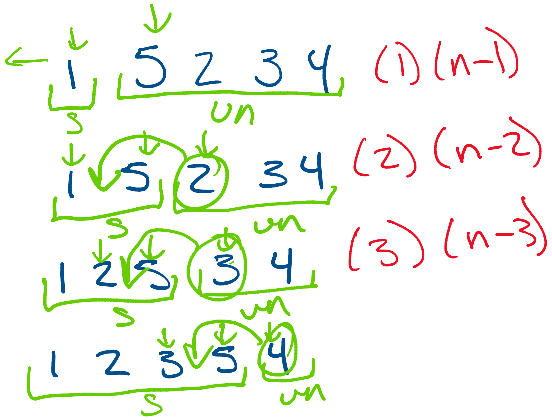
Runtime:

Worst-case: $O(n^2)$ → essentially, have to swap each element w/ every other element
5 4 3 2 1
$(n-1)n \approx n^2$

Best-case: $O(n)$
→ already sorted

1 2 3 4 5

→ Insertion Sort
→ Sorted & unsorted sections
→ for each element in unsorted, find where it belongs in sorted & "insert"



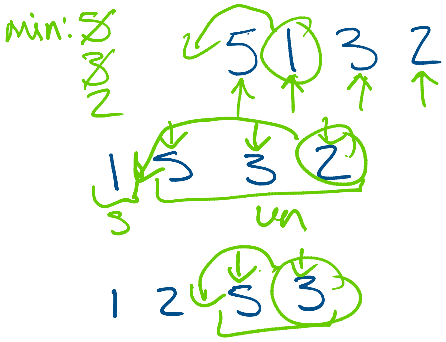
Run time:

→ worst-case: $O(n^2)$
→ for every element, going through rest of list

→ best-case: $O(n)$
→ already sorted

→ already sorted

→ Selection Sort
- → Sorted & unsorted
- → repeatedly find smallest in unsorted & add to end of sorted section
  - → LL: insert node after last in sorted
  - → arrays:
    - → insertions → shifting O(n)
    - → swap first in the unsorted section w/ smallest element O(1)



Runtime:
- → worst-case: $O(n^2)$
  - → loop through entire list n times
    1^{st}, 2^{nd}, 3^{rd}, ... n^{th}
- → best-case: $O(n^2)$ *
  - → already sorted