Video Name: ArrayList

Topics:

- ArrayList syntax
- Parameterized vs unparameterized
- Autoboxing

Java Class(es): ArrayList

declare unparameterized (heterogeneous):

```
ArrayList things = new ArrayList();
```

can add different types of objects:

```
things.add("Cyndi");
things.add(new Point(5,6));
```

iterated for loop – very useful!

```
for (Object o : things)
   System.out.println(o);
```

comparable for loop (disadvantage: have to tell it how many)

```
for (int i=0; i<2; i++)
   System.out.println(things.get(i));
```

must use typecast if want to access element as a specific type – error prone!

```
Point getPoint = (Point) things.get(1);
```

declare parameterized (safer, no need for type casts):

```
ArrayList<Point> points;
points = new ArrayList<Point>();
```

store and access:

```
points.add(new Point(3,4));
Point p = points.get(0);
```

iterated for loop (see that we can use Point function directly, since parameterized list)

```
for (Point p2 : points)
   System.out.println(p2.getX());
```

Can remove items (other items “slide” so there are no gaps)

```
points.remove(0);
```

ArrayList must contain Object, not primitives

```
ArrayList<int> is not valid. Must use wrapper class.
ArrayList<Integer> numbers = new ArrayList<Integer>();
```

Java will “auto box” – automatically wrap value in Wrapper class

```
numbers.add(12);
```

If you have a lot of items, using a wrapper class is not efficient. Better to use array.

```
int[] bigData = new int[100000];
```