ANONYMOUS CLASSES, LAMBDAS, COLLECTIONS

INNER CLASSES

- Sometimes, we just need a class for one small purpose
- Standard option: private inner class ButtonListener
    - Quite a bit of code
    - Potentially hard to read

ANONYMOUS CLASSES

- A class without a name
- define class directly where it's needed and instantiated
- better, but still quite a bit of code

LAMBDA EXPRESSIONS

- A lot of times, the purpose of class is to implement simple interface with one method
- Example `Comparable`, `ActionListener`, etc.
- Because interface has only one function
    - No need to explicitly specify
    - It can infer

LAMBDA EXPRESSIONS

- Compact way of passing around behavior
- Lambda functions allow for brief, clean implementation of interface with one function
- Make code easier to read

LAMBDAS - FORMAT

- Single expression, one argument:

```
ActionListener oneArgHello = event -> System.out.println("hello")
```

- Multiple expressions, enclose expressions in bracket
- No arguments, use empty `()` for arg (aka in place of event)
- Multiple arguments, use `(x,y)` for arg

COLLECTIONS

- Group of objects (elements)
- Gathers/organizes elements
- Interface in Java - specifies how to interact with (access/manage) group of objects
- An abstraction - details of implementation hidden

COLLECTIONS

- Some are ordered, others unordered
- Some allow duplicates, others don't
- Example: `ArrayList` implements `Collection`
  - Implementation differs from other implementers of `Collection`
  - Stil interact in the same way