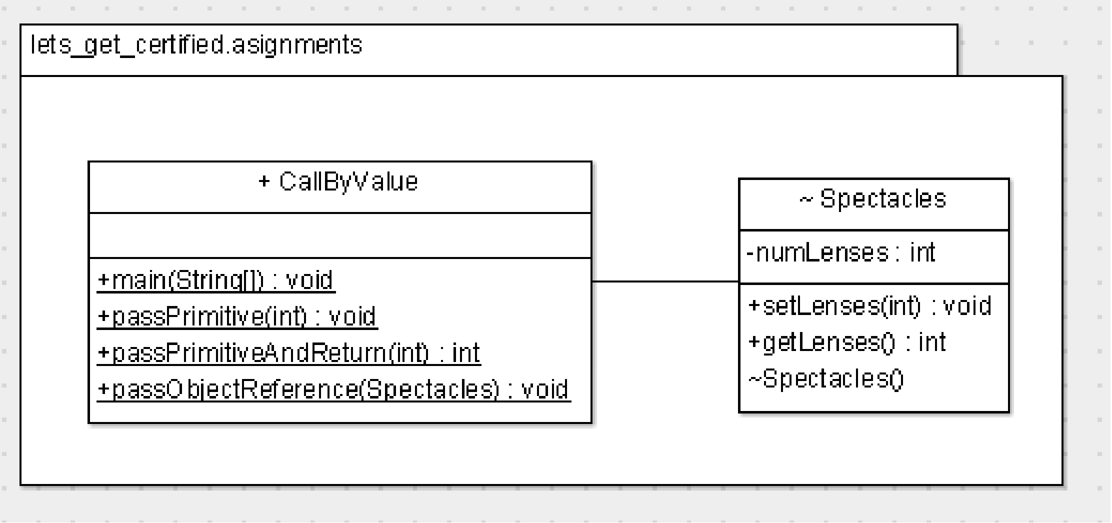
CallByValue exercise


lets_get_certified.asignments
+ CallByValue
+main(String[]) : void
+passPrimitive(int) : void
+passPrimitiveAndReturn(int) : int
+passObjectReference(Spectacles) : void
~ Spectacles
-numLenses : int
+setLenses(int) : void
+getLenses() : int
~Spectacles()

1. Passing primitives:
   a. *passPrimitive()* method:
      i. set the parameter passed in to 50
   b. *passPrimitiveAndReturn()* method:
      i. set the parameter passed in to 50 and return it
   c. in *main()*:
      i. declare a local *int* variable ‘x’ and initialise it to 10
      ii. output *x*
      iii. invoke *passPrimitive()* passing down *x*
      iv. output *x* – note how it’s value **has NOT** changed
      v. invoke *passPrimitiveAndReturn()* passing down *x*. Assign the return of the method call into *x* i.e. overwrite *x* with the return value.
      vi. output *x* – note how it’s value **has** changed

2. Passing object references:
   a. code the *Spectacles* class as per the UML
      i. the constructor sets the instance variable to 2
      ii. the ‘set’ and ‘get’ methods operate as expected i.e. change and retrieve the instance variable respectively.
   b. *passObjectReference()* method:
      i. using the reference passed in, invoke *setLenses(1)*
   c. in *main()*:
      i. create an instance of *Spectacles*, refer to it using a reference named ‘specs’
      ii. output the number of lenses using *specs*
      iii. invoke *passObjectReference()*, passing down *specs*
      iv. output the number of lenses using *specs* – note how it’s value **has** changed