# Effective Java, 2010

## Exercises

1. **Static factory methods, Object methods**. Create Point.java class that contains 2 final int fields x and y.
   1. Create factory methods
      * Takes x and y parameters
      * Only takes x, sets y to 0
      * Only takes y, sets x to 0
      * Takes no parameters, produces (0,0) point
   2. Reuse the same Point instance any time (0,0) point is requested
   3. Make Point serializable and implement readResolve
   4. Implement equals correctly
   5. Implement hashCode correctly
   6. Make Point Cloneable and implement clone correctly
   7. Make Point Comparable and implement compareTo
2. **Builder.** Create Polygon class that holds array of Points. Create PolygonBuilder with point(int x, int y) method that builds the Polygon according to the pattern we learned. Use provided Builder interface.
3. **Numeric overflows.** Implement ReverseComparator that takes a Comparator as parameter and returns the opposite result.
   1. Create an ExtremeNaturalOrderComparator, that returns Integer.MIN\_VALUE, 0, and Integer.MAX\_VALUE only
   2. Sort a list using ExtremeNaturalOrderComparator
   3. Sort the list using your ReverseComparator, if it does not work – fix it.
4. **Decorator using interfaces.** Create InstrumentedHashSet class that extends HashSet and counts additions to the set by overriding add and addAll methods.
   1. Test addAll – the additions are counted twice.
   2. Instead extending HashSet, extend ForwardingSet provided to you, sending new HashSet in the constructor
   3. Test addAll - now it works as expected
   4. Improve the class even further – rename it to InstrumentedSet, take Set in the constructor and pass it on to ForwardingSet
   5. Test with both TreeSet and HashSet
5. **Class initialization**. What is wrong with Elvis? Can you fix it?
6. **Generics.** Take HeapTemplate and add generics transforming the class to Heap – the definitions need to be as flexible as possible and the code has to compile, implementation is not important.
7. **Generics – THC.** Go over Registry implementation – Service Locator pattern. Note Class.cast usage
8. **Multi-threading.** Implement a thread-safe BoundedSet, a set that can hold at most CAPACITY elements.
9. **Annotations+Multithreading+THC+Builder+Varargs**. AutoRegistry is an enhanced Registry which is capable of loading default implementation for service interface, if no implementation was registered. Use AutoRegistryTemplate, implement AnnotationBuilder and service initialization using ConcurrentMap.putIfAbsent. Test with RegistryTest.