```
package genericCheckpointing.driver;

import genericCheckpointing.util.ProxyCreator;

// import the other types used in this file

public class Driver {

    public static void main(String[] args) {

        // FIXME: read the value of checkpointFile from the command line

        ProxyCreator pc = new ProxyCreator();

        // create an instance of StoreRestoreHandler (which implements
        // the InvocationHandler

        // create a proxy
        StoreRestoreI cpointRef = (StoreRestoreI) pc.createProxy(
                                                                 new Class[] {
                                                                     StoreI.class,
RestoreI.class
                                                                 },
                                                                 new StoreRestoreHandler()
                                                                 );

        // FIXME: invoke a method on the handler instance to set the file name for
checkpointFile and open the file

        MyAllTypesFirst myFirst;
        MyAllTypesSecond  mySecond;

        // Use an if/switch to proceed according to the command line argument
        // For deser, just deserliaze the input file into the data structure and then print
the objects
        // The code below is for "serdeser" mode
        // For "serdeser" mode, both the serialize and deserialize functionality should be
called.

        // create a data structure to store the objects being serialized
        // NUM_OF_OBJECTS refers to the count for each of MyAllTypesFirst and
MyAllTypesSecond
        for (int i=0; i<NUM_OF_OBJECTS; i++) {

            // FIXME: create these object instances correctly using an explicit value
constructor
            // use the index variable of this loop to change the values of the arguments to
these constructors
            myFirst = new MyAllTypesFirst(...);
            mySecond = new MyAllTypesSecond(..);

            // FIXME: store myFirst and mySecond in the data structure
            ((StoreI) cpointRef).writeObj(myFirst, "XML");
            ((StoreI) cpointRef).writeObj(mySecond, "XML");

        }

        SerializableObject myRecordRet;

        // create a data structure to store the returned ojects
        for (int j=0; j<2*NUM_OF_OBJECTS; j++) {

            myRecordRet = ((RestoreI) cpointRef).readObj("XML");
            // FIXME: store myRecordRet in the vector
        }

        // FIXME: invoke a method on the handler to close the file (if it hasn't already been
closed)

        // FIXME: compare and confirm that the serialized and deserialzed objects are equal.
        // The comparison should use the equals and hashCode methods. Note that hashCode
        // is used for key-value based data structures

    }
}
```