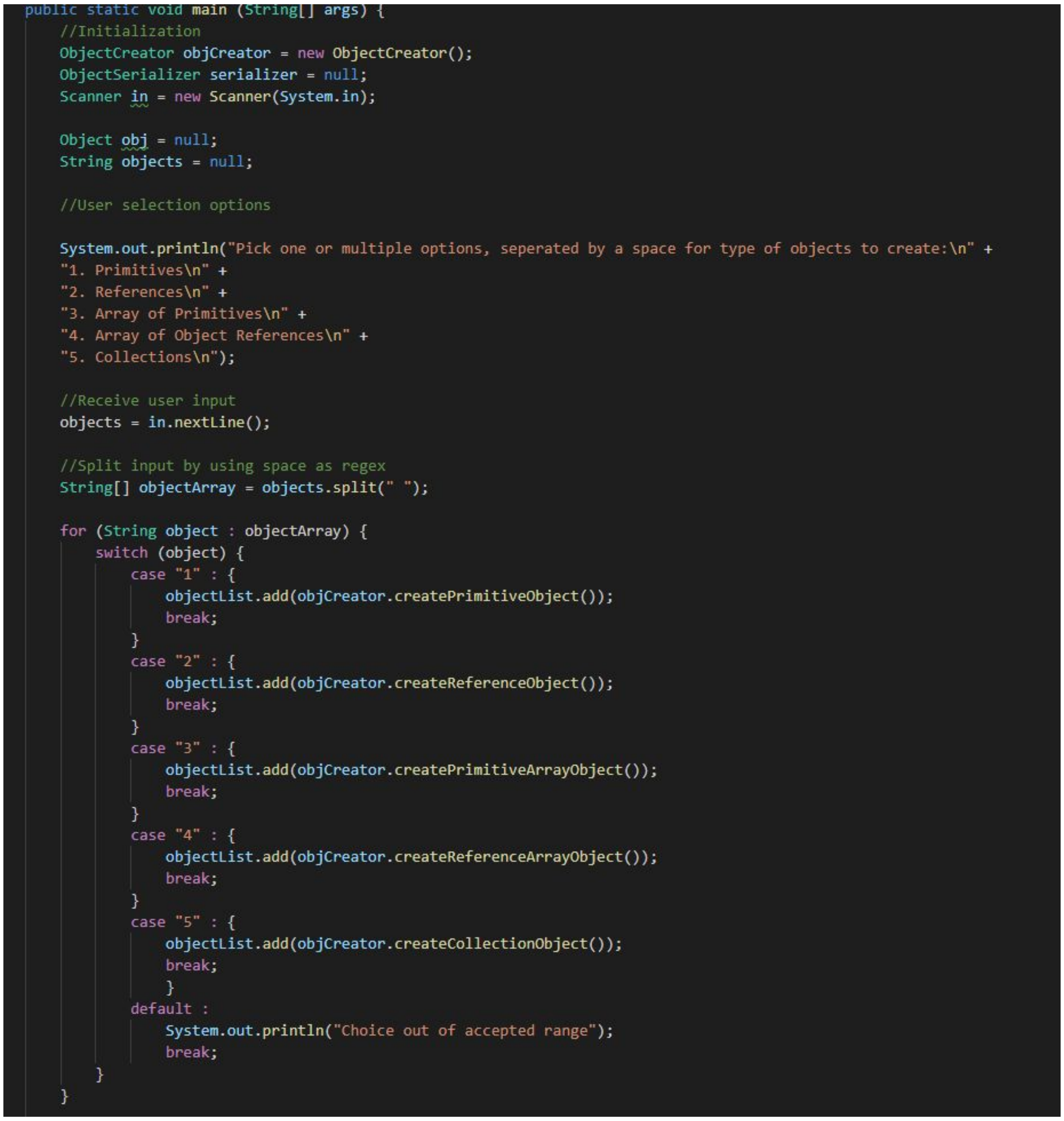
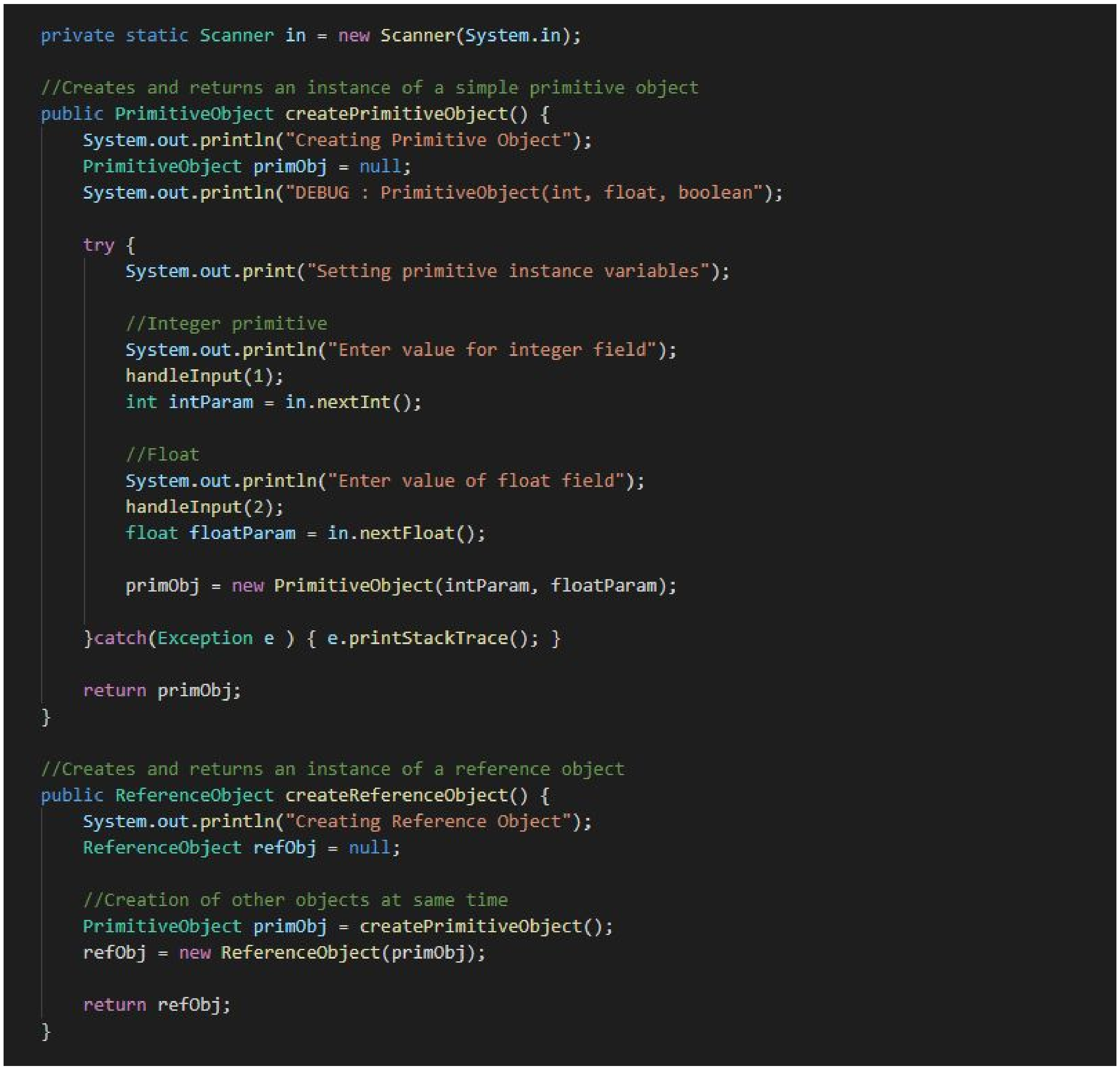
REFACTORING



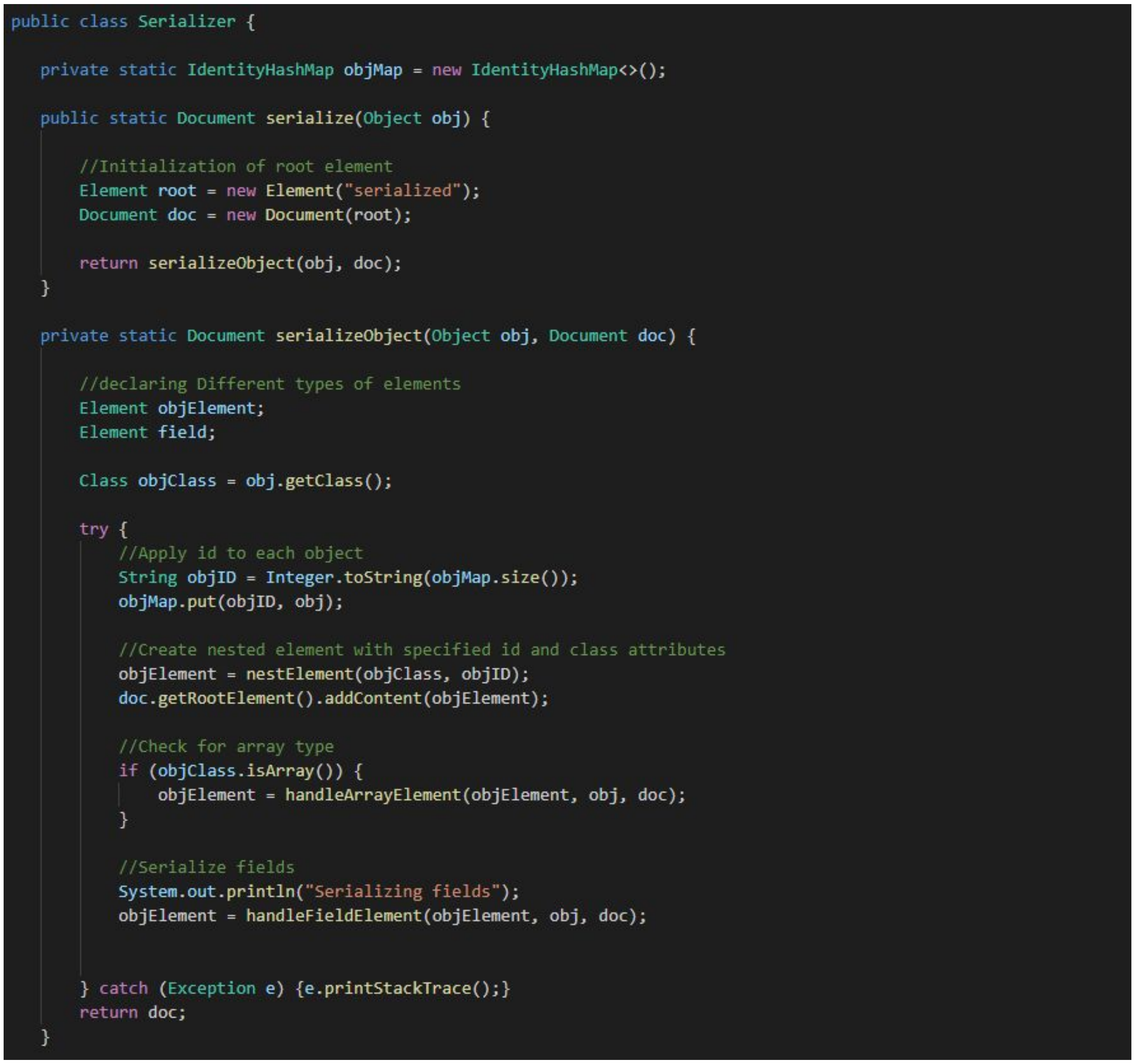
```
public static void main (String[] args) {
    //Initialization
    ObjectCreator objCreator = new ObjectCreator();
    ObjectSerializer serializer = null;
    Scanner in = new Scanner(System.in);

    Object obj = null;
    String objects = null;

    //User selection options

    System.out.println("Pick one or multiple options, seperated by a space for type of objects to create:\n" +
    "1. Primitives\n" +
    "2. References\n" +
    "3. Array of Primitives\n" +
    "4. Array of Object References\n" +
    "5. Collections\n");

    //Receive user input
    objects = in.nextLine();

    //Split input by using space as regex
    String[] objectArray = objects.split(" ");

    for (String object : objectArray) {
        switch (object) {
            case "1" : {
                objectList.add(objCreator.createPrimitiveObject());
                break;
            }
            case "2" : {
                objectList.add(objCreator.createReferenceObject());
                break;
            }
            case "3" : {
                objectList.add(objCreator.createPrimitiveArrayObject());
                break;
            }
            case "4" : {
                objectList.add(objCreator.createReferenceArrayObject());
                break;
            }
            case "5" : {
                objectList.add(objCreator.createCollectionObject());
                break;
                }
            default :
                System.out.println("Choice out of accepted range");
                break;
        }
    }
```

Limited work done by sender by dividing the work done into a helper class called Object Creator.



```
private static Scanner in = new Scanner(System.in);

//Creates and returns an instance of a simple primitive object
public PrimitiveObject createPrimitiveObject() {
    System.out.println("Creating Primitive Object");
    PrimitiveObject primObj = null;
    System.out.println("DEBUG : PrimitiveObject(int, float, boolean");

    try {
        System.out.print("Setting primitive instance variables");

        //Integer primitive
        System.out.println("Enter value for integer field");
        handleInput(1);
        int intParam = in.nextInt();

        //Float
        System.out.println("Enter value of float field");
        handleInput(2);
        float floatParam = in.nextFloat();

        primObj = new PrimitiveObject(intParam, floatParam);

    }catch(Exception e ) { e.printStackTrace(); }

    return primObj;
}

//Creates and returns an instance of a reference object
public ReferenceObject createReferenceObject() {
    System.out.println("Creating Reference Object");
    ReferenceObject refObj = null;

    //Creation of other objects at same time
    PrimitiveObject primObj = createPrimitiveObject();
    refObj = new ReferenceObject(primObj);

    return refObj;
}
```

Divided the work done in the object creation into multiple methods instead of just a few.



```java
public class Serializer {

    private static IdentityHashMap objMap = new IdentityHashMap<>();

    public static Document serialize(Object obj) {

        //Initialization of root element
        Element root = new Element("serialized");
        Document doc = new Document(root);

        return serializeObject(obj, doc);
    }

    private static Document serializeObject(Object obj, Document doc) {

        //declaring Different types of elements
        Element objElement;
        Element field;

        Class objClass = obj.getClass();

        try {
            //Apply id to each object
            String objID = Integer.toString(objMap.size());
            objMap.put(objID, obj);

            //Create nested element with specified id and class attributes
            objElement = nestElement(objClass, objID);
            doc.getRootElement().addContent(objElement);

            //Check for array type
            if (objClass.isArray()) {
                objElement = handleArrayElement(objElement, obj, doc);
            }

            //Serialize fields
            System.out.println("Serializing fields");
            objElement = handleFieldElement(objElement, obj, doc);


        } catch (Exception e) {e.printStackTrace();}
        return doc;
    }
```

Created the parent Serializer class which is called by the object Serializer class which takes part of the methods and overrides the others to suit it’s needs.