```
/**Supports Array related operations like get(index),
put(value),remove, length, toString and fromString methods **/

import java.util.ArrayList;
import java.util.List;

import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.annotate.JsonAutoDetect.Visibility;
import org.codehaus.jackson.annotate.JsonMethod;
import org.codehaus.jackson.map.ObjectMapper;

public class ArrayFormat {

      private List<Object> arrayData;

      public ArrayFormat() {
            arrayData = new ArrayList<Object>();
      }

      public ArrayFormat(ArrayList<Object> arrayData) {
            this.arrayData = arrayData;
      }

      public ArrayFormat put(String value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            this.arrayData.add(value);
            return this;
      }

      public ArrayFormat put(Integer value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            this.arrayData.add(value);
            return this;
      }

      public ArrayFormat put(Double value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            this.arrayData.add(value);
            return this;
      }

      public ArrayFormat put(ArrayFormat value) {
```

```
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            this.arrayData.add(value);
            return this;
      }

      public ArrayFormat put(ObjectFormat value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            this.arrayData.add(value);
            return this;
      }

      public String getString(int index) throws
ArrayIndexOutOfBoundsException, DataTypeMisMatchException {
            if (index > (arrayData.size() - 1)) {
                  throw new ArrayIndexOutOfBoundsException("Invalid
index");
            }
            String value = null;
            try {
                  value = (String) arrayData.get(index);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of string type");
            }

            return value;
      }

      public int getInt(int index) throws
ArrayIndexOutOfBoundsException, DataTypeMisMatchException {
            if (index > (arrayData.size() - 1)) {
                  throw new ArrayIndexOutOfBoundsException("Invalid
index");
            }
            int value;
            try {
                  value = (int) arrayData.get(index);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of integer type");
            }
            return value;
      }

      public Double getDouble(int index) throws
ArrayIndexOutOfBoundsException, DataTypeMisMatchException {
```

```
            if (index > (arrayData.size() - 1)) {
                  throw new ArrayIndexOutOfBoundsException("Invalid
index");
            }
            Double value = null;
            try {
                  value = (Double) arrayData.get(index);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of double type");
            }

            return value;
      }

      public ArrayFormat getArray(int index) throws
ArrayIndexOutOfBoundsException, DataTypeMisMatchException {
            if (index > (arrayData.size() - 1)) {
                  throw new ArrayIndexOutOfBoundsException("Invalid
index");
            }
            ArrayFormat value = null;
            try {
                  value = (ArrayFormat) arrayData.get(index);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of array type");
            }
            return value;
      }

      public ObjectFormat getObject(int index) throws
ArrayIndexOutOfBoundsException, DataTypeMisMatchException {
            if (index > (arrayData.size() - 1)) {
                  throw new ArrayIndexOutOfBoundsException("Invalid
index");
            }
            ObjectFormat value = null;
            try {
                  value = (ObjectFormat) arrayData.get(index);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of type object");
            }
            return value;
      }

      public Object get(int index) throws
ArrayIndexOutOfBoundsException {
            if (index > (arrayData.size() - 1)) {
                  throw new ArrayIndexOutOfBoundsException("Invalid
index");
```

```
            }
            return arrayData.get(index);
      }

      public int length() {
            return arrayData.size();
      }

      public Object remove(int index) {
            if (index > (arrayData.size() - 1))
                  return null;
            return arrayData.remove(index);
      }

      public String toString() {
            String jsonInString = null;
            ObjectMapper mapper = new ObjectMapper();
            mapper.setVisibility(JsonMethod.FIELD,
Visibility.ANY);

            try {
                  jsonInString =
mapper.writeValueAsString(this.arrayData);
            } catch (Exception e) {
                  e.printStackTrace();
            }
            return jsonInString;
      }

      @SuppressWarnings("unchecked")
      public static ArrayFormat fromString(String
stringRepresentation) {
            ObjectMapper mapper = new ObjectMapper();
            JsonNode value = null;
            ArrayList<Object> arrayFromString;
            try {
                  value = mapper.readTree(stringRepresentation);
            } catch (Exception e) {
                  e.printStackTrace();
            }
            arrayFromString = mapper.convertValue(value,
ArrayList.class);
            ArrayFormat array = new ArrayFormat();
            array.arrayData = arrayFromString;
            return array;
      }

}
```