```
/** supports Object related operations, put, get, remove, length,
toString and fromString **/

import java.util.Hashtable;

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 ObjectFormat {

      private Hashtable<String, Object> map;

      public ObjectFormat() {
            map = new Hashtable<String, Object>();
      }

      public ObjectFormat(Hashtable<String, Object> value) {
            map = value;
      }

      public ObjectFormat put(String key, String value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            this.map.put(key, value);
            return this;

      }

      public ObjectFormat put(String key, Integer value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            map.put(key, value);
            return this;
      }

      public ObjectFormat put(String key, Double value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            map.put(key, value);
            return this;
      }

      public ObjectFormat put(String key, ArrayFormat value) {
```

```
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            map.put(key, value);
            return this;
      }

      public ObjectFormat put(String key, ObjectFormat value) {
            if (value == null) {
                  throw new IllegalArgumentException("Value cannot
be null");
            }
            map.put(key, value);
            return this;
      }

      public String getString(String key) throws
DataTypeMisMatchException {
            if (!map.containsKey(key)) {
                  throw new IllegalArgumentException("key not
found");
            }
            String value = null;
            try {
                  value = (String) map.get(key);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of string type");
            }

            return value;
      }

      public Integer getInt(String key) throws
DataTypeMisMatchException {
            if (!map.containsKey(key)) {
                  throw new IllegalArgumentException("key not
found");
            }
            Integer value = null;
            try {
                  value = (Integer) map.get(key);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of integer type");
            }
            return value;
      }

      public Double getDouble(String key) throws
DataTypeMisMatchException {
```

```
            if (!map.containsKey(key)) {
                  throw new IllegalArgumentException("key not
found");
            }
            Double value = null;
            try {
                  value = (Double) map.get(key);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of double type");
            }
            return value;
      }

      public ArrayFormat getArray(String key) throws
DataTypeMisMatchException {
            if (!map.containsKey(key)) {
                  throw new IllegalArgumentException("key not
found");
            }
            ArrayFormat value = null;
            try {
                  value = (ArrayFormat) map.get(key);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of array type");
            }
            return value;
      }

      public ObjectFormat getObject(String key) throws
DataTypeMisMatchException {
            if (!map.containsKey(key)) {
                  throw new IllegalArgumentException("key not
found");
            }
            ObjectFormat value = null;
            try {
                  value = (ObjectFormat) map.get(key);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of object type");
            }
            return value;
      }

      public Object get(String key) {
            if (!map.containsKey(key)) {
                  throw new IllegalArgumentException("key not
found");
            }
            return map.get(key);
```

```
	}

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

	public Object remove(String key) {
		if (!map.containsKey(key))
			return null;
		return map.remove(key);
	}

	public String toString() {
		String jsonInString = null;
		ObjectMapper mapper = new ObjectMapper();
		mapper.setVisibility(JsonMethod.FIELD,
Visibility.ANY);
		try {
			jsonInString =
mapper.writeValueAsString(this.map);
		} catch (Exception e) {
			e.printStackTrace();
		}
		return jsonInString;
	}

	@SuppressWarnings("unchecked")
	public static ObjectFormat fromString(String
stringRepresentation) {
		ObjectMapper mapper = new ObjectMapper();
		JsonNode value = null;
		Hashtable<String, Object> objectFromString;
		try {
			value = mapper.readTree(stringRepresentation);
		} catch (Exception e) {
			e.printStackTrace();
		}
		objectFromString = mapper.convertValue(value,
Hashtable.class);
		ObjectFormat map = new ObjectFormat();
		map.map = objectFromString;
		return map;
	}

}
```