```
/** Cursor holds a value from the database, current state of the
value will be known to the cursor
 * Observers of Cursor will be notified when the cursor value
changes using notifyObserver
 **/

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

public class Cursor implements ICursor {
      private Object value;
      private IDatabase database;
      String key;
      private List<ICursorObserver> observers = new ArrayList<>();

      public Cursor(IDatabase database, String key) {
            this.database = database;
            this.value = database.data().get(key);
            this.key = key;
      }

      @Override
      public Object value() {
            this.value = database.data().get(key);
            if (this.value != null &&
this.value.getClass().isArray()) {
                  return Arrays.deepToString((Object[])
this.value);
            }
            return this.value;
      }

      @Override
      public void notifyObserver() {
            if (observers != null && !observers.isEmpty()) {
                  for (ICursorObserver observer : observers) {
                        observer.update(this.key, value());
                  }
            }
      }

      @Override
      public void addObserver(ICursorObserver observer) {
            this.observers.add(observer);
      }

      @Override
      public void removeObserver(ICursorObserver observer) {
            this.observers.remove(observer);
      }
```

```
	@Override
	public Object get() {
		return database.data().get(this.key);
	}

	@Override
	public int getInt() throws DataTypeMisMatchException {
		int value;
		try {
			value = (int) database.data().get(this.key);
		} catch (ClassCastException e) {
			throw new DataTypeMisMatchException("value is not
of int type");
		}
		return value;
	}

	@Override
	public double getDouble() throws DataTypeMisMatchException {
		double value;
		try {
			value = (double) database.data().get(this.key);
		} catch (ClassCastException e) {
			throw new DataTypeMisMatchException("value is not
of double type");
		}
		return value;
	}

	@Override
	public ArrayFormat getArray() throws
DataTypeMisMatchException {
		ArrayFormat value = null;
		try {
			value = (ArrayFormat)
database.data().get(this.key);
		} catch (ClassCastException e) {
			throw new DataTypeMisMatchException("value is not
of array type");
		}
		return value;
	}

	@Override
	public String getString() throws DataTypeMisMatchException {
		String value;
		try {
			value = (String) database.data().get(this.key);
		} catch (ClassCastException e) {
			throw new DataTypeMisMatchException("value is not
of string type");
```

```
            }
            return value;
      }

      @Override
      public ObjectFormat getObject() throws
DataTypeMisMatchException {
            ObjectFormat value = null;
            try {
                  value = (ObjectFormat)
database.data().get(this.key);
            } catch (ClassCastException e) {
                  throw new DataTypeMisMatchException("value is not
of type object");
            }
            return value;
      }
}
```