```java
import java.io.IOException;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.regex.Pattern;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.client.Get;
import org.apache.hadoop.hbase.client.HBaseAdmin;
import org.apache.hadoop.hbase.client.HTable;
import org.apache.hadoop.hbase.client.HTableInterface;
import org.apache.hadoop.hbase.client.Increment;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.log4j.Logger;

public class HbaseUtils {
	private static Logger logger = Logger.getLogger(HbaseUtils.class);
	private static Configuration conf = null;
	static int count = 0;

	/**  根据行键进行数据查询
	  * @param tableName  表名称,现有表有：sql_gdm_m03_item_sku_da_jss_201405
	  * @param rowKey  行键
	  * @param columns  返回列过滤(item_first_cate_cd,item_second_cate_cd,
	  *              item_third_cate_cd,work_post_cd,shop_id,dept_id_1,
	  *              dept_name_1,dept_id_2,dept_name_2,dept_id_3,dept_name_3)
	  */
	public static Map<String, String> getResult(HTableInterface table, String cf, String rowKey, String... columns)
throws IOException {
		Get get = new Get(Bytes.toBytes(rowKey));
		for (String column : columns) {
			get.addColumn(Bytes.toBytes(cf), Bytes.toBytes(column));
		}
		Map<String, String> map = new HashMap<String, String>();
		Result result = table.get(get);
		List<KeyValue> list = result.list();
		if (list != null) {
			for (KeyValue kv : result.list()) {
				map.put(Bytes.toString(kv.getQualifier()), Bytes.toString(kv.getValue()));
```

```
            }
        }
        return map;
    }

    /*   遍历查询 hbase 表
     * @tableName  表名
     */
    public static void getResultScann(String tableName, String start_rowkey, String stop_rowkey) throws
IOException {
        Scan scan = new Scan();
        scan.setStartRow(Bytes.toBytes(start_rowkey));
        scan.setStopRow(Bytes.toBytes(stop_rowkey));
        ResultScanner rs = null;
        HTable table = new HTable(conf, Bytes.toBytes(tableName));
        try {
            rs = table.getScanner(scan);
            for (Result r : rs) {
                for (KeyValue kv : r.list()) {
                    logger.error("row:" + Bytes.toString(kv.getRow()));
                    logger.error("family:" + Bytes.toString(kv.getFamily()));
                    logger.error("qualifier:" + Bytes.toString(kv.getQualifier()));
                    logger.error("|" + Bytes.toString(kv.getValue()));
                    logger.error("timestamp:" + kv.getTimestamp());

                }
            }
        } finally {
            rs.close();
            table.close();
        }
    }

    /*  删除表   @tableName  表名
     */
    public static void deleteTable(String tableName) throws IOException {
        HBaseAdmin admin = new HBaseAdmin(conf);
        admin.disableTable(tableName);
        admin.deleteTable(tableName);
        admin.close();
        System.out.println(tableName + "is deleted!");
    }

    /**  获取表名称   */
    public static String loadTable() {
        HBaseAdmin admin = null;
        logger.debug("++++++++++++++++++++++++++++++start");
        try {
            String prefix = "sql_gdm_m03_item_sku_da_jss_";
            admin = new HBaseAdmin(conf);
            Pattern p = Pattern.compile("^" + prefix + ".\\d+$");
            String[] tables = admin.getTableNames(p);
            if (tables == null || tables.length == 0) {
                return null;
            }

            int[] suffixs = new int[tables.length];
            for (int i = 0; i < tables.length; i++) {
                logger.debug("----------table Name-------------" + tables[i]);
                String suffix = tables[i].substring(tables[i].lastIndexOf("_") + 1);
                suffixs[i] = Integer.parseInt(suffix);
            }
            Arrays.sort(suffixs);
            logger.debug("+++++++++++++++++++++++++++++end");
            return prefix + suffixs[suffixs.length - 1];
        } catch (Exception ex) {
            logger.error("获取表异常：", ex);
            return null;
        } finally {
            try {
                admin.close();
            } catch (IOException io) {
                logger.error("关闭表异常：", io);
            }
        }
    }

    /*  创建表
      * @tableName  表名
      * @family  列族列表
      */
    public static void creatTable(String tableName, String[] family) throws Exception {
        HBaseAdmin admin = new HBaseAdmin(conf);
        HTableDescriptor desc = new HTableDescriptor(tableName);
        for (int i = 0; i < family.length; i++) {
            desc.addFamily(new HColumnDescriptor(family[i]));
        }
        if (admin.tableExists(tableName)) {
            logger.error("---table " + tableName + " Exists!---");
            System.exit(0);
        } else {
            admin.createTable(desc);
            logger.debug("---create table " + tableName + " Success!---");
        }
        admin.close();
    }

    public static void IncrValue(HTableInterface htable, String rowKey, String family, String columnArr[], long
valueArr[]) {
        try {
            htable.setAutoFlush(false);
            Increment inc = new Increment(Bytes.toBytes(rowKey));
            for (int j = 0; j < columnArr.length; j++) {
                inc.addColumn(Bytes.toBytes(family), Bytes.toBytes(columnArr[j]), valueArr[j]);
            }
            htable.increment(inc);
        } catch (Exception e) {
            logger.error("IncrValue error msg", e);
        }

    }

    /**
     * @param tableName
     * @param familyColumn
     * @param rowKey
     * @param columnArr
     * @param time
     * @param valueArr
     */
    public static boolean addData(HTableInterface htable, String familyColumn, String rowKey, String columnArr[],
long ts, String valueArr[])
            throws IOException {
        Put put = null;
        try {
            htable.setAutoFlush(false);
            put = new Put(Bytes.toBytes(rowKey));
            for (int j = 0; j < columnArr.length; j++) {
                put.add(Bytes.toBytes(familyColumn),            Bytes.toBytes(columnArr[j]),            ts,
Bytes.toBytes(valueArr[j]));
            }
            htable.put(put);
            return true;
        } catch (Exception e) {
            logger.error("error count=" + count++);
            return false;
        }
    }

    public static String getRowKey(long orderId) {
        String id = String.valueOf(orderId);
        if (id == null) {
            return null;
        }
        StringBuffer buf = new StringBuffer(id).reverse();
        while (buf.length() < 13) {
            buf.append("0");
        }
        return buf.toString();
    }

    /**  字符串左补零  */
    public static String getLeftAddZero(String str, int len) {
        len = len - str.length();
        for (int i = 0; i < len; i++) {
            str = "0" + str;
        }
        return str;
    }

    /**  查询 hbase 中的 sku 的信息   */
    public static Map<String, String> getCityParComRELA(HTableInterface htable) {
        ResultScanner rs = null;
        Map<String, String> map = new HashMap<String, String>();
        try {
            Scan scan = new Scan();
            scan.addColumn(Bytes.toBytes("f"), Bytes.toBytes("centerNum"));
            rs = htable.getScanner(scan);
            if (rs != null) {
                for (Result r = rs.next(); r != null; r = rs.next()) {
                    for (KeyValue kv : r.raw()) {
                        map.put(Bytes.toString(kv.getRow()), Bytes.toString(kv.getValue()));
                    }
                }
            }

        } catch (Exception ex) {
            logger.error("hbase query getCityParComRELA error", ex);
        }
        return map;
    }

    public static void main(String[] args) {
        System.out.println(getRowKey(120876300L));
    }
}
```