Item.java

```
package org.com;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "item_details")

public class Item {

	@Id

	@Column(name = "item_id")

	private int id;

	@Column(name = "item_name")

	private String name;

	@Column(name = "item_qty")

	private int qty;


	static int pocductCount;

	static {

		Item.pocductCount = 0;

	}


	public Item(int id, String name, int qty) {

		super();

		Item.pocductCount++;
```

```
        this.id = id;
        this.name = name;
        this.qty = qty;
    }
    public Item(){
        }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getQty() {
        return qty;
    }
    public void setQty(int qty) {
        this.qty = qty;
    }
}
```

Solution.java

```
package org.com;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class Solution {

	BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));

	SessionFactory sf = new Configuration().configure().buildSessionFactory();

	Session session = sf.openSession();

	public void create() throws NumberFormatException, IOException {

		session.beginTransaction();

		System.out.println("Enter  the id : ");

		int id = Integer.valueOf(bf.readLine());

		System.out.println("Enter  the name : ");

		String name = bf.readLine();

		System.out.println("Enter  the quantity : ");

		int quantity = Integer.valueOf(bf.readLine());

		Item obj = new Item(id, name, quantity);

		session.save(obj);

		session.getTransaction().commit();

	}

	public void update() throws NumberFormatException, IOException {

		int flag = 1, flag2 = 1;
```

```
                System.out.println("1.For  update \n 2.Not update");
                flag = Integer.valueOf(bf.readLine());
                while (flag == 1) {
                        System.out.println("Enter  the id which want to be update : ");
                        int id = Integer.valueOf(bf.readLine());
                        Item obj = session.get(Item.class, id);
                        session.beginTransaction();
                        while (flag2 == 1) {
                                System.out.println("Enter  which one want to update 1. for name
        2. for quantity\n");
                                int choice = Integer.valueOf(bf.readLine());
                                switch (choice) {
                                case 1: {
                                        System.out.println("Enter  the name : ");
                                        String name = bf.readLine();
                                        obj.setName(name);
                                        break;
                                }
                                case 2: {
                                        System.out.println("Enter  the quantity : ");
                                        int quantity = Integer.valueOf(bf.readLine());
                                        obj.setQty(quantity);
                                        break;
                                }
                                default:
                                        break;
                                }
```

```
                    System.out.println("1.to  update another column \n 2.Not
continue");
                    flag2 = Integer.valueOf(bf.readLine());
                }
                session.update(obj);
                session.save(obj);
                session.getTransaction().commit();
                System.out.println("1.For  update \n 2.Not update");
                flag = Integer.valueOf(bf.readLine());
            }
        }
        public void delete() throws NumberFormatException, IOException {
            int flag = 1, flag2 = 1;
            System.out.println("1.For  delete \n 2.Not update");
            flag = Integer.valueOf(bf.readLine());
            while (flag == 1) {
                System.out.println("Enter  the id which want to be update : ");
                int id = Integer.valueOf(bf.readLine());
                Item obj = session.get(Item.class, id);
                session.beginTransaction();
                session.delete(obj);
                session.save(obj);
                session.getTransaction().commit();
                System.out.println("1.For  delete \n 2.Not delete");
                flag = Integer.valueOf(bf.readLine());
            }
        }
```

```
	public void search() throws  NumberFormatException,  IOException {
		int flag = 1, flag2 = 1;
		System.out.println("1.For  search \n 2.Not search");
		flag = Integer.valueOf(bf.readLine());
		while (flag == 1) {
			System.out.println("Enter  the id which want to be search : ");
			int id = Integer.valueOf(bf.readLine());
			Item obj = session.get(Item.class, id);
			if (obj != null) {
				session.beginTransaction();
				System.out.println("The  Id : " + obj.getId());
				System.out.println("The  name : " + obj.getName());
				System.out.println("The  quantity  : " + obj.getQty());
				session.getTransaction().commit();
			}
			System.out.println("1.To  search \n 2.Not continue");
			flag = Integer.valueOf(bf.readLine());
		}
	}
	public static void main(String[] args) throws NumberFormatException,  IOException {
		BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
		int choice = 1;
		Solution ob = new Solution();
		while (choice >= 1 && choice <= 4) {
			System.out.println("1.  For create \n 2. For update \n 3. For delete \n 4.
For search \n 5.exit");
			choice = Integer.valueOf(bf.readLine());
```

```
                        switch (choice) {
                        case 1: {
                                    ob.create();
                                    break;
                        }
                        case 2: {
                                    ob.update();
                                    break;
                        }
                        case 3:{
                                    ob.delete();
                                    break;
                        }
                        case 4:{
                                    ob.search();
                                    break;
                        }
                        default : break;
                        }
            }
      }
}
```

Hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ~ Hibernate, Relational Persistence for Idiomatic Java ~ ~ License:
      GNU Lesser General Public License (LGPL), version 2.1 or later. ~ See the
```

```
        lgpl.txt file in the root directory or <http://www.gnu.org/licenses/lgpl-2.1.html>.  -->
<!DOCTYPE hibernate-configuration PUBLIC
      "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
      "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <!-- Database connection settings -->
                <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
                <property
name="connection.url">jdbc:mysql://localhost:3306/sample</property>
                <property name="connection.username">root</property>
                <property name="connection.password"></property>
                <!-- JDBC connection pool (use the built-in) -->
                <property name="connection.pool_size">10</property>
                <!-- SQL dialect -->
                <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
                <!-- Disable the second-level cache -->
                <property
name="cache.provider_class">org.hibernate.cache.internal.NoCacheProvider</property>
                <!-- Echo all executed SQL to stdout -->
                <property name="show_sql">true</property>
                <!-- Drop and re-create the database schema on startup -->
                <property name="hbm2ddl.auto">update</property>
                <!-- Names the annotated entity class -->
                        <!-- <mapping class="com.emp.sample.Employee" /> -->
                        <!-- <mapping resource="com/emp/sample/employee.hbm.xml"  /> -->
                <mapping class="org.com.Item" />
                        <!-- <mapping resource="com/book/book.hbm.xml"  /> -->
        </session-factory>
</hibernate-configuration>
```