**HashMap custom key object example**

In this example, I have created an Account class with only two fields for simplicity. I have overridden the hashcode and equals method such that it uses only account number to verify the uniqueness of Account object. All other possible attributes of Account class can be changed on runtime.

|  |
| --- |
| package com.howtodoinjava.demo.map;  public class Account  {      private int accountNumber;      private String holderName;        public Account(int accountNumber) {          this.accountNumber = accountNumber;      }        public String getHolderName() {          return holderName;      }        public void setHolderName(String holderName) {          this.holderName = holderName;      }        public int getAccountNumber() {          return accountNumber;      }        //Depends only on account number      @Override      public int hashCode() {          final int prime = 31;          int result = 1;          result = prime \* result + accountNumber;          return result;      }        //Compare only account numbers      @Override      public boolean equals(Object obj) {          if (this == obj)              return true;          if (obj == null)              return false;          if (getClass() != obj.getClass())              return false;          Account other = (Account) obj;          if (accountNumber != other.accountNumber)              return false;          return true;      }    } |

Will this cause any undesired behavior???

NO, it will not. The reason is that Account class’s implementation honor the contract that “Equal objects must produce the same hash code as long as they are equal, however unequal objects need not produce distinct hash codes.” i.e.

1. Whenever a.equals(b) is true, then a.hashCode() must be same as b.hashCode().
2. Whenever a.equals(b) is false, then a.hashCode() may/may not be same as b.hashCode().

**Test the HashMap custom key object**

Lets test our Account class for above analysis.

|  |
| --- |
| TestMutableKey.java |
| package com.howtodoinjava.demo.map;  import java.util.HashMap;  public class TestMutableKey  {      public static void main(String[] args)      {          //Create a HashMap with mutable key          HashMap<Account, String> map = new HashMap<Account, String>();            //Create key 1          Account a1 = new Account(1);          a1.setHolderName("A\_ONE");          //Create key 2          Account a2 = new Account(2);          a2.setHolderName("A\_TWO");            //Put mutable key and value in map          map.put(a1, a1.getHolderName());          map.put(a2, a2.getHolderName());            //Change the keys state so hash map should be calculated again          a1.setHolderName("Defaulter");          a2.setHolderName("Bankrupt");            //Success !! We are able to get back the values          System.out.println(map.get(a1)); //Prints A\_ONE          System.out.println(map.get(a2)); //Prints A\_TWO            //Try with newly created key with same account number          Account a3 = new Account(1);          a3.setHolderName("A\_THREE");            //Success !! We are still able to get back the value for account number 1          System.out.println(map.get(a3)); //Prints A\_ONE      }  } |

Program Output.

|  |
| --- |
| Console |
| A\_ONE  A\_TWO  A\_ONE |