Answer: The ***removeDuplicates*** method used HashMap to check duplicate employees. The issue here is the side effect of using mutable fields in the hashCode() method of Employee. Since the Employee object used as key in hashMap is mutated (tracker.get(e).setVisited(**true**)) on subsequent check next time, the ***containsKey*** method fails between employee having same **name** and ***salary*** as the ***visited*** values are different. This means the hashcode generated are different in this case.

It’s necessary to use immutable objects as keys in a hashtable to avoid such issues.

Solution:

@Override

**public** **boolean** equals(Object ob) {

**if**(ob == **null**) **return** **false**;

**if**(!(ob **instanceof** Employee)) **return** **false**;

Employee emp = (Employee)ob;

**return** emp.name.equals(name) && emp.salary == salary ;

//&& emp.visited == visited; removed mutable fields from equals

}

@Override

**public** **int** hashCode() {

**int** result = 17;

**long** longval = Double.*doubleToLongBits*(salary);

**int** salaryHash = (**int**) (longval ^ (longval >>> 32));

result += 31 \* result + name.hashCode();

result += 31 \* result + salaryHash;

//result +=31 \* result + (visited ? 1:0); removed mutable fields from hashCode

**return** result;

}

//setter removed

// public void setName(String name) {

// this.name = name;

// }

// public void setSalary(int salary) {

// this.salary = salary;

// }