`Employee e1` does a set of projects. `Employee e2` also does all the projects did by `e1` except the first project, in place of which `e2` does another project. Write a program that defines two classes `Employee` and `Test`. Define copy constructor to create `e2` from `e1` in such a way that changing the values of instance variables of either `e2` or `e1` should not affect the other one. The code takes name of `e2` and new project done by `e2` as input. Complete the program as specified below.

- Class `Employee` that has the following members.
  - Private instance variables `String name` and `String[] projects` to store name and projects respectively
  - Define required constructor(s)
  - Accessor methods `getName( )` and `getProject( )` to get name of employee and project at specific index.
  - Mutator methods `setName( )` and `setProject( )` to set name of employee and project at specific index.
- Class `Test` that has the method `main` which does the following.
  - Two objects of `Employee e1` and `e2` are created. `e2` is created using `e1`
  - name of `e2` and second item bought by `c2` are updated by taking the input
  - name of `e1`, `e2` and first project done by `e1` and `e2` are printed

Ans :

```
import java.util.*;
class Employee{
   String name;
   String[] projects;
   public Employee(String n, String[] proj){
      name = n;
      projects = proj;
   }
   public Employee(Employee e){
      this.name = e.name;
      int l = e.projects.length;
      this.projects = new String[l];
      for(int i = 0; i < l; i++){
         this.projects[i] = e.projects[i];
      }
   }
   public void setName(String n) {
      name = n;
   }
   public void setProject(int index, String proj) {
      projects[index] = proj;
```

```
    }
    public String getName() {
        return name;
    }
    public String getProject(int indx) {
        return projects[indx];
    }
}
public class Test {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String[] proj =  {"PJ1", "PJ2", "PJ3"};
        Employee e1 = new Employee("Surya", proj);
        Employee e2 = new Employee(e1);
        e2.setName(sc.next());
        e2.setProject(0, sc.next());
        System.out.println(e1.getName() + ": " + e1.getProject(0));
        System.out.println(e2.getName() + ": " + e2.getProject(0));
    }
}
```

Write the Java code as instructed.

- Class `Faculty` has the following members.
  - `String name`, `double salary` as private instance variables
  - Constructor to initialize the instance variables
  - Method `public double bonus(float percent)` that returns the bonus computed as (percent/100.0)*salary
  - You should define method `getDetails()` to display `name` and `salary` of an `Faculty`
  - You should overload method `getDetails()` as `getDetails(float percent)` to display the `name`, `salary` and `bonus` of an `Faculty`
- Class `Hod` extends class `Faculty` and has the following members.
  - `String personalAssistant` as private instance variable
  - Constructor to initialize the instance variables of `Hod` that includes the instance variables of `Faculty`
  - You should override method `public double bonus(float percent)` that returns the bonus of a `Hod` as 50 percent less bonus than a regular faculty
  - You should override method `getDetails()` to display the `name`, `salary` and `personalAssistant` of `Hod`.
  - You should override method `getDetails(float percent)` to display the `name`, `salary`, `personalAssistant` and `bonus` of a `Hod`

```
import java.util.Scanner;
class Faculty{
   private String name;
   private double salary;
   public Faculty(String name, double salary) {
      this.name = name;
      this.salary = salary;
   }
   public double bonus(float percent){
      return (percent/100.0)*salary;
   }
    public String getDetails() {
       return name + ", " + salary;
    }
    public String getDetails(float percent ) {
       return getDetails()+ ", bonus = "+bonus(percent);
    }
}
class Hod extends Faculty{
    private String personalAssistant;
    public Hod(String name, double salary, String pa) {
       super(name, salary);
       this.personalAssistant = pa;
    }
    public double bonus(float percent){
       return 0.5*super.bonus(percent);
    }
    public String getDetails() {
       return super.getDetails()+", "+ personalAssistant;
    }
    public String getDetails(float percent ) {
       return getDetails()+", "+bonus(percent);
    }
}
public class InheritanceTest{
    public static void main(String[] args) {
       Scanner sc=new Scanner(System.in);
       Faculty obj1 = new Faculty(sc.next(), sc.nextDouble());
       Faculty obj2 = new Hod(sc.next(), sc.nextDouble(), sc.next());
       System.out.println(obj1.getDetails());
       System.out.println(obj1.getDetails(10));
       System.out.println(obj2.getDetails());
       System.out.println(obj2.getDetails(10));
    }
```

}

Write Java code as instructed.

- Define an interface `Appraisable` that has the following members:
  - Default method `default void appraisal(Teacher t)` that increments the `salary` of an `Employee` by `(stuPassPer/100)*5000`.
  - Abstract method `public abstract void checkAndUpdateSalary()`
- Define an interface `SpecialAppraisable` that extends `Appraisable` and has the following members:
  - Default method `default void spAppraisal(Teacher t)` that increments the salary of an `Employee` by `(stuPassPer/100)*10000`.
- Class `Teacher` that implements the interface `SpecialAppraisable` and has the following members:
  - `String name`, `double salary` and `private double stuPassPer` as private instance variables
  - Constructor to initialize the instance variables
  - Mutator method to update `salary`
  - Accessor method to access `salary`
  - Accessor method to access `stuPassPer`
  - Override method `toString()` that returns `name`, `salary` and `stuPassPer` of the `Teacher` as a single concatenated string (each separated by a single space)
  - Overriding method `public void checkAndUpdateSalary()` that has the following functionality.
    - If `stuPassPer>=60 and stuPassPer<75` then invoke the `appraisal()` method from `Appraisable` interface
    - Else, if `stuPassPer>=75 and stuPassPer<=100` then invoke the`spAppraisal()` method from SpecialAppraisable interface
- Class `InterfaceTest` that has the following members:
  - You should define method `public static void printUpdatedTeachList(Teacher[] tList)` that has the following functionality
    - Iterate over array `tList` and invoke method `checkAndUpdateSalary()` on each `Teacher` object.
    - Then, iterate over `tList` and display each `Teacher` object.
  - `main` method has the following functionality
    - Creates and initializes an array `tArr` of three `Teacher` objects
    - Invokes method `printUpdatedTeachList(Teacher[] tList)` to print the updated details of each `Teacher` after the appraisal is applied

```
import java.util.*;
interface Appraisable{
   default void appraisal(Teacher t){
      t.setSalary(t.getSalary()+(t.getstuPassPer()/100)*5000);
   }
   public abstract void checkAndUpdateSalary();
}
```

```
interface SpecialAppraisable extends Appraisable{
    default void spAppraisal(Teacher t){
        t.setSalary(t.getSalary()+(t.getstuPassPer()/100)*10000);
    }
}
class Teacher implements SpecialAppraisable{
    private String name;
    private double salary;
    private double stuPassPer;
    public Teacher(String name, double salary, double stuPassPer) {
        this.name = name;
        this.salary = salary;
        this.stuPassPer = stuPassPer;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public double getstuPassPer() {
        return stuPassPer;
    }
    public String toString() {
        return name + ", " + salary + ", " + stuPassPer;
    }
    public void checkAndUpdateSalary() {
        if(stuPassPer >= 60 && stuPassPer < 75)
            appraisal(this);
        else if(stuPassPer >= 75 && stuPassPer <= 100)
            spAppraisal(this);
     }
}
public class InterfaceTest {
    public static void printUpdatedTeachList(Teacher[] tList) {
        for (int i = 0; i < tList.length; i++)
            tList[i].checkAndUpdateSalary();
        for (int i = 0; i < tList.length; i++)
            System.out.println(tList[i]);
    }
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
        Teacher tArr[] = new Teacher[3];
        for (int i = 0; i < tArr.length; i++)
```

```
            tArr[i] = new Teacher(sc.next(), sc.nextDouble(), sc.nextDouble());
        InterfaceTest.printUpdatedTeachList(tArr);
    }
}
```

Write a Java program that takes as input 4 Shop objects and the list of `Shop` objects with attributes shop `name`, and number of items sold `nsold`. The program should add each customer name as key and number of items as value to the map object. After adding all objects to the map, display the shop name which has sold maximum number of items as shown in the test cases. Complete the program as specified below:

- Class `Shop` that has the following members:
  - `String name`, `int nsold` as private instance variable
  - Constructor to initialize the `name` and `nsold`
  - Accessor methods to all instance variables
- Class `MapTest` has the following members:
  - You should define method `public static void printShopName(ArrayList<Shop> sList)` that does the following:
    - Iterates over array `sList` such that in each iteration, add each customer name as key and number of items as value to the map object.
    - Print the shop name which has sold maximum number of items.
  - `main` method has the following functionality
    - Creates a list of 4 `Shop` objects.
    - Invokes method `printShopName(list)` to print the shop name which has sold maximum number of items.

```
import java.util.*;
class Shop{
    private String name;
    private int nsold;
    public Shop(String s, int ns){
            this.name = s;
            this.nsold = ns;
    }
    public String getName(){
            return name;
    }
    public int getItemSold(){
            return nsold;
    }
}
public class MapTest {
```

```
        public static void printShopName(ArrayList<Shop> sList) {
            Map<String, Integer> m = new LinkedHashMap<String, Integer>();
            String shop = "";
            int sold = 0;
for(Shop s: sList)
    m.put(s.getName(), m.getOrDefault(s.getName(),0)+s.getItemSold());

for (HashMap.Entry<String, Integer> entry : m.entrySet()){
   if(entry.getValue()> sold) {
         shop = entry.getKey();
      sold = entry.getValue();
   }
}
      System.out.println(shop+" : "+sold);
   }
   public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
         ArrayList<Shop> list = new ArrayList<Shop>();
         for (int i = 0; i < 4; i++) {
                  list.add(new Shop(sc.next(), sc.nextInt()));
         }
         printShopName(list);
   }
}
```