```java
public interface Order {    ……. 1

	public void displayOrderDetails(); ……. 0.5
	public double getPrice();    ……. 0.5

}



public class Dish {

	private String name;
	private double price;

	public Dish(String s, double p) {
		name = s;
		price = p;
	}

	public double getPrice() throws Exception {
		if (price < 0 || price > 100) throw new Exception("Wrong price");
		return price;
	}

	public void display() {
		System.out.println(name + price);
	}
}
```

```
public abstract class Meal implements Order {    ……. 1 + 1    ……. /14

	private String name;
	private Dish[] arrDish;  ……. 1
	private int nbDish;   ……. 1

	public Meal(String s, int size) { ……. /2
		name = s;
		arrDish = new Dish[size];   ……. 1
		nbDish = 0;   ……. 1
	}

	public double totalDishPrice() {   ……. /6
		double t = 0.0;  ……. 1

		for (int i=0; i < nbDish; i++) { ……. 1
			try {  ……. 1
				t += arrDish[i].getPrice();  ……. 1
			}
			catch(Exception e) { ……. 1
				System.out.println(e.getMessage());
			}
		}

		return t; ……. 1
	}

	public void displayOrderDetails() {  ……. /2
		for (int i=0; i < nbDish; i++)  ……. 1
			arrDish[i].display();  ……. 1
	}
}

public class DineIn extends Meal {

	private int tableNo;

	public DineIn(String name, int size, int tn) {
		super(name, size);
		tableNo = tn;
	}

	public double getPrice() {
		return this.totalDishPrice() * 1.05;
	}
}
```

```java
public class HomeDelivery extends Meal {  ……. 1          ……. /14

	private String address;
	private int distance;

	public HomeDelivery(String name, int size, String adr, int d) {  ……. /2
		super(name ,size); ……. 1
		address = adr;      ……. 0.5
		distance = d;       ……. 0.5
	}

	public double calculateDeliveryFeesRecursive(int d) {  ……. /3
		if (d <= 10) ……. 1
			return 5.0;  ……. 1
		else
			return 1.05 * calculateDeliveryFeesRecursive(d -1); ……. 1
	}

	public double calculateDeliveryFeesIterative(int d) {      ……. /3
		double fee = 5.0;   ……. 0.5

		for (int i = 11; i < d; i++)     ……. 1
			fee = fee * 1.05;   ……. 1

		return fee;  ……. 0.5
	}

	public double getPrice() {       ……. /4
		double p;

		p = totalDishPrice() + calculateDeliveryFees(distance); ……. 1 + 1 + 1

		return p; ……. 1
	}

	public int getDistance() {……. /1
		return distance;
	}
}
```

```
public class Menu {        ……. /23
	private String name;
	private Meal[] arrMeal; ……. 1
	private int nbMeal; ……. 1

	public Menu(String s, int size) {       ……. /2
		name = s;
		arrMeal = new Meal[size]; ……. 1
		nbMeal = 0;  ……. 1
	}

	public Meal mostExpensive() {    ……. /5
		Meal res = arrMeal[0];    ……. 1
		for (int i = 1; i < nbMeal; i++) {     ……. 1
			if (arrMeal[i].getPrice() > res.getPrice())   ……. 1
				res = arrMeal[i]; ……. 1
		}
		return res; ……. 1
	}

	public Meal secondMostExpensive() {    ……. /5
		Meal most = arrMeal[0];   ……. 0.5
		Meal second = null;       ……. 0.5

		for (int i = 1; i < nbMeal; i++) {     ……. 0.5
			if (arrMeal[i].getPrice() > most.getPrice()) {       ……. 0.5
				second = most;      ……. 0.5
				most = arrMeal[i];  ……. 0.5
			}
			else {
				if (   second == null || ……. 0.5
					arrMeal[i].getPrice() > second.getPrice())   ……. 0.5

						second = arrMeal[i];      ……. 0.5
			}
		}

		return second;      ……. 0.5
	}
```

```
public void split(HomeDelivery[] arHL, DineIn[] arDI, int d, double p)   ……. /9
             throws Exception {  ……. 0.5
             int j = 0, k=0;     ……. 0.5 + 0.5

             for (int i=0; i < nbMeal; i++) { ……. 0.5
                   if (arrMeal[i] instanceof HomeDelivery) { ……. 0.5
                       if (((HomeDelivery) arrMeal[i]).getDistance() == d) { 0.5+0.5
                                 if (j < arHL.length) ……. 0.5
                                  arHL[j++] = (HomeDelivery) arrMeal[i]; …0.5+0.5+0.5
                                 else
                                       throw new ……. 0.5
                                       Exception("Number of HomeDeliv. exceeded!");
                            }
                   }
                   else {
                         if (arrMeal[i].getPrice() >= p) {       ……. 0.5
                                 if (k < arDI.length)       ……. 0.5
                                   arDI[k++] = (DineIn) arrMeal[i]; ……. 0.5+0.5+0.5
                                 else
                                       throw new ……. 0.5
                                       Exception("Number of Dine-In exceeded!");

                            }
                   }
             }
       }

}
```