```java
public interface Finance {

	public double calculateBudget();
	public void display();

}

public abstract class Ministry implements Finance{

	private String name;
	protected int nbEmployee;

	public Ministry(String name, int nbEmployee) {
		this.name = name;
		this.nbEmployee = nbEmployee;
	}

	public Ministry(Ministry m){
		this.name = m.name;
		this.nbEmployee = m.nbEmployee;
	}

	public String getName() {
		return name;
	}

	public int getNbEmployee() {
		return nbEmployee;
	}

	public void display(){
		System.out.println("Name: " + name);
		System.out.println("nbEmployee: " + nbEmployee);
	}
}

public class Executive extends Ministry{

	private double expenses;

	public Executive(String name, int nbEmployee, double expenses) {
		super(name, nbEmployee);
		this.expenses = expenses;
	}

	public Executive(Executive e){
		super(e);
		this.expenses = e.expenses;
	}

	public double getExpenses() {
		return expenses;
	}
	public double calculateBudget(){
		return expenses + nbEmployee * 1.5;
	}


	@Override
	public void display(){
		super.display();
		System.out.println("Expenses: " + expenses);
	}
}



public class Administrative extends Ministry{

	private double balance;

	public Administrative(String name, int nbEmployee, double balance) {
		super(name, nbEmployee);
		this.balance = balance;
	}

	public Administrative(Administrative a){
		super(a);
		this.balance = a.balance;
	}

	public double getBalance() {
		return balance;
	}

	public double calculateBudget(){
		return nbEmployee * 10000 - balance;
	}

	@Override
	public void display(){
		super.display();
		System.out.println("Balance: " + balance);
	}
}

public class Other extends Ministry{

	public Other(String name, int nbEmployee) {
		super(name, nbEmployee);
	}
	public Other(Other o){
		super(o);
	}
	public double calculateBudget(){
		return nbEmployee * 10000;
	}
}
```

```java
public class Government {

	private String name;
	private Ministry arMins[];
	private int nbMins;

	public Government(String name, int size) {
		this.name = name;
		arMins = new Ministry[size];
		nbMins = 0;
	}

	public void addMinistry(Ministry m){
		if(nbMins >= arMins.length) System.out.println("The array is full");
		else{
			if(m instanceof Executive)
				arMins[nbMins++] = new Executive((Executive) m);
			else if(m instanceof Administrative)
				arMins[nbMins++] = new Administrative((Administrative) m);
			else if(m instanceof Other)
				arMins[nbMins++] = new Other((Other) m);
			System.out.println("Ministry was added successfully");
		}
	}

	public double averageOfBudget(){
		double sum = 0;
		for(int i = 0; i < nbMins; i++)
			sum += arMins[i].calculateBudget();
		return sum / nbMins;
	}

	public int countExecutives(double e){
		int count = 0;
		for(int i = 0; i < nbMins; i++)
			if(arMins[i] instanceof Executive &&
			((Executive)arMins[i]).getExpenses() > e)
				count++;
		return count;
	}

	public int getExecutives(double e, Executive ae[]){
		int count = 0;
		for(int i = 0; i < nbMins; i++){
			if(arMins[i] instanceof Executive &&
((Executive)arMins[i]).getExpenses() > e &&
		((Executive)arMins[i]).calculateBudget() > averageOfBudget())
				ae[count++] = (Executive) arMins[i];
		}
		return count;
	}

}
```