school of computing, informatics, & decision systems engineering

Ira A. Fulton Schools of Engineering
Arizona State University

CSE 460: Software Analysis and Design

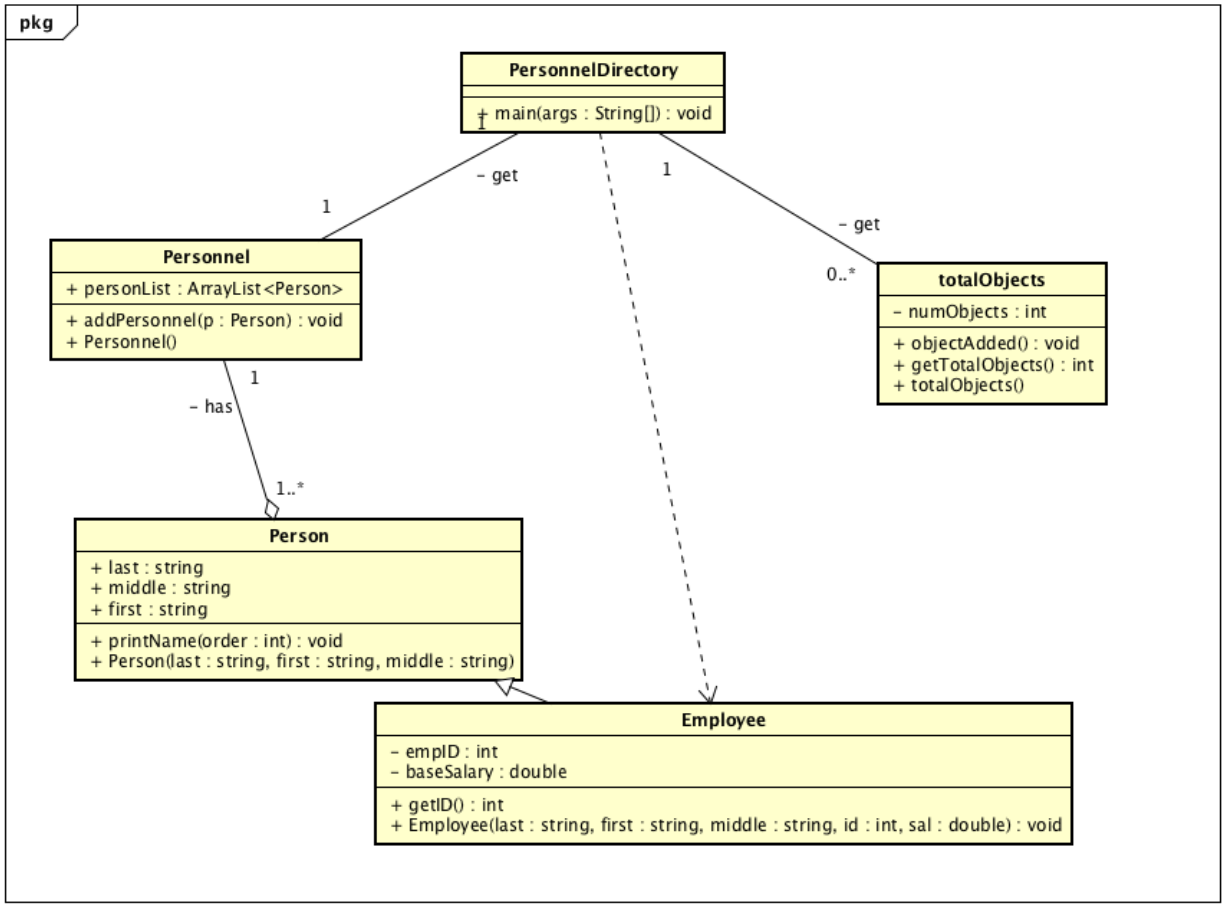
Directory Management System Phase I Submission

Directions:

Complete your work for Phase I Parts 1 and 2 in this document. Save and submit as a **single**PDF titled “Last Name_First Name_Directory Management System Project_Phase I_Submission”.

Phase I, Part 1

Use the Astah tool to draw the class diagram for the current implementation of the university system. Use correct UML notations. When you have completed the diagram, take a clear screenshot and paste it in the space provided.



Phase I, Part 2

In the code, identify object-oriented concept violations, content coupling, common coupling, control coupling and stamp coupling situations. Copy and paste the code segments that shows each coupling situation in the space provided. *You may use additional space as necessary.*

1. **Object-Oriented Concept Violations**

[paste code segments here]

```
public class Person {
        public String last;
        public String first;
        public String middle;

public ArrayList<Person> personList;
```

These violate Encapsulation concepts.

```
public class totalObjects
{

  private static int numObjects = 0;

  public totalObjects()
  {
    numObjects=0;
  }

  public void objectAdded()
  {
    numObjects++;
  }

  public int getTotalObjects()
  {
    return numObjects;
  }
}
public int getID()
        {
                return empID;

        }
```

These are Data Abstraction violations.

How would you fix these violations?

I would fix the Encapsulation violations by making the public variables in the Person class and Personnel class private. For the Data Abstraction violations I would remove the totalObjects class and just use a personList.size() in a method to get the total number of objects and I would remove the getID() method because it is never used.

2. **Content Coupling**

[paste code segments here]

```
boolean found = false;
int loc =-1;
for(int i =0; i <per.personList.size(); i++)
{
	if( per.personList.get(i).first.equals(firstN) && per.personList.get(i).last.equals(lastN))
	{
		found = true;
		loc = i;
		}
	}

	if(found)
	{
		System.out.println("Found");
		per.personList.get(loc).printName(0);

	}else
	{
		System.out.println("not found");
		Person p1  = new Person(lastN, firstN, " ");
		per.personList.add(p1);
		total.objectAdded();
	}
```

How would you fix this?

I would fix this by calling addPersonnel and not per.personList.add to reduce the coupling and that the PersonnelDirectory is more interested in the responsibilities of the interface.

3. **Common Coupling**

[paste code segments here]

```
  private static int numObjects = 0;
else
	{
		System.out.println("not found");
		Person p1  = new Person(lastN, firstN, " ");
		per.personList.add(p1);
		total.objectAdded();
	}
```

How would you fix this?

[Write your answer in this space.]

Having a static variable is like having a global variable and by removing this and the totalObject class I can fix the common coupling error that could happen with this variable and the PersonnelDirectory class is modifiying the public personList from Personnel class so I would move this into the Personnel class to reduce coupling.

4. **Control Coupling**

[paste code segments here]

```
public void printName(int order)
        {

          if(order == 0)
          {
             System.out.println(first + "  " + middle + "  " + last);

          }else if(order == 1)
             {

             System.out.println(last + " ," + middle + " " + first);

             }
             else if(order == 2)
                              {

                              System.out.println(last + " ," + first + " " + middle);

             }
        }
System.out.println("Enter the order 0: first, middle,  last, 1: first, last, middle, 2: last, first , middle
");
                              int order = scan.nextInt();
                              for(int i=0; i<per.personList.size(); i++)
                              {

                                       per.personList.get(i).printName(order);
                              }
```

How would you fix this?

[Write your answer in this space.]

I would fix this by creating three different print functions in the Person class so that it does not rely on the int order as a control parameter from PersonnelDirectory class.

5. **Stamp Coupling**

```
[paste code segments here]
public void addPersonnel(Person p)
        {
                personList.add(p);
        }
```