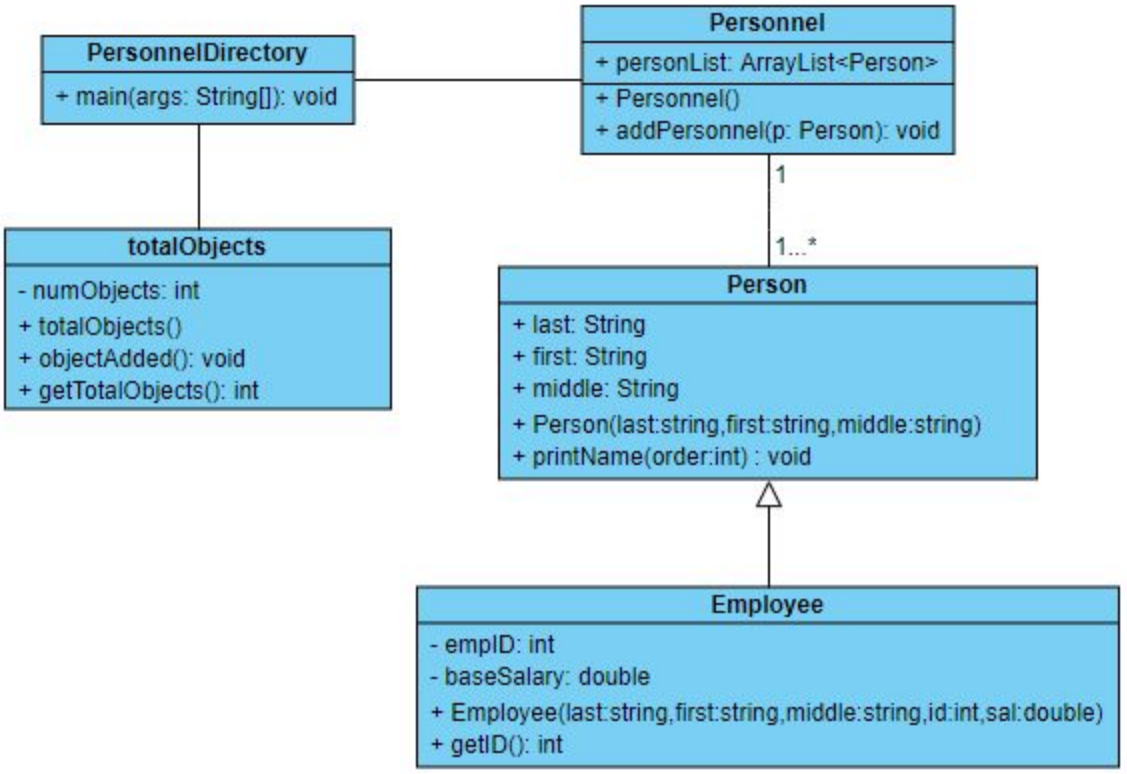
Jacob Peters
CSE464 - HW6

Part 1:

1)

a)



b)

Content Coupling:

From PersonnelDirectory:

```
per.addPersonnel(e1);
```

From Personnel:

```
public Personnel() {
        personList = new ArrayList<Person>();
}
```

Common Coupling:

From Employee:

```
super(last, first, middle);
```

Control Coupling:

From PersonnelDirectory:

```
Employee e1  = new Employee(lastN, firstN, middleN, empID, salary);
per.addPersonnel(e1);
total.objectAdded();
```

Stamp Coupling:

From PersonnelDirectory:

```
per.personList.get(loc).printName(0);
```

c) Basically, remove any of the dependencies from the smaller classes that only serve to hold values. I.e. no need for the personnel class since it can be made into an ArrayList in the directory class and no need for the totalObjects class since it just is a counter. And instead of having both a person and employee class, just combine them together.
In the “HW6_1.zip”

Part 2:

1)

a)

The essence of something being modular means that it can run essentially by itself. So having low coupling and high cohesion means that it is linked through code to as little other functions as possible and does as much of the needed computations for its output itself.

b)

Integration Type	About	Pros	Cons
Top Down	Works by using stubs to simulate the lower level calls.	Helps find problems at the top of the code. Isn’t very complex.	Not very useful if the topside of the code is defect free.
Bottom Up	Using test drivers, data is passed to lower level modules that would be further down the code. And once they are verified they are applied to higher levels and repeat.	Finding problems at the bottom of the code. Works on smaller components then works its way to larger ones.	Much more complex than top down.
Sandwich	Combination of the other two approaches. The code is broken down into layers and when	Useful in larger projects with many stubs. Very time efficient. Provides a larger	Cannot be used with systems that have high coupling dependencies between layers.

	testing you focus on running tests for the adjacent layer of where you are currently testing and build up more layers as you go.	coverage of testing stubs.	Very high use of stubs and drivers.

c)

Framework: A pre-setup control-flow where the actual space for user input is predefined and readily usable in the initial state with just inserting their own cod into.

Toolkit: A toolkit however is a type of library that can be used to execute code in the client.

d)

Annotation	Use	Example
@Before	Run before the @Test	Give an initial state for a test
@After	Run after the @Test is completed	Can be used to check the final state after a test
@BeforeClass	Run once before any of the @Test	Used to setup the system for all the tasks
@AfterClass	Run after all the @Test are completed	Used after all the tests are run and can be used to show stats of the runs or similar data items.

2)

a) Note, in the actual code the speeds are 4,6,12 not 6,12,20

Test #	Calls	Final Speed
1	dialUp	0
2	leverUp	4
3	leverUp,dialUp,	6
4	dialUp,dialUp,leverUp	12
5	leverUp,leverUp	30

6	leverUp,leverUp,leverUp	60
7	leverUp,leverDown	0

b)

windshieldWiperTest.java

c)

i)

Test #	Change	Speed
1	Lever up	4
2	Dial Up	6
3	Dial Up	12
4	Lever Up	30
5	Lever Down	12
6	Lever Down	0

ii)

windshieldWiperTest2.java