



Lab 8 –Inheritance

Exercise 1

Create a file named A8.java. Place all your code in this file.

Write an interface called IPerson. Add a method signature for payMoney. It should return an int and accept an int as a parameter.

Exercise 2

Write a class called Person that implements IPerson. You should define a private field named money of type int. You must implement the payMoney method. Given input integer toPay, your method should decrease money by toPay and return the new value of money. Add a default constructor that sets money to 10000 (10000 = \$100.00).

Override toString and display your money (e.g. if money = 12345, display \$123.45). In this assignment, you should always use the override annotation when overriding a method.

Note: Make sure the money is always displayed with 2 decimal places (so 10000 should show as \$100.00, not \$100.0 and not \$100). You can use the String.format method as follows: String.format("%.2f", amount)

Override equals to compare the values in money.

Write a getter for money.

Exercise 3

Write a class called `Employee` that extends `Person`. Add a private integer field called `employeeID`. Write a new value constructor that accepts as input an integer called `employeeID`. Your constructor should call `Person`'s constructor so that `money` is still initialized (how can you do this?). Set the field `employeeID` to the parameter `employeeID`. These should have the same name.

Override the `payMoney` method. Make sure you use the override annotation. Employees only need to pay half. Before subtracting the amount to pay from `money`, divide `toPay` by 2, rounding up.

Hint: How can you access the parent's `payMoney`?

Hint: The `Math` class has a method that can help with rounding up.

Override `toString` and display your `employeeID` and `money` (e.g. if `employeeID` = 12 and `money` = 100, print out "Employee 12 has \$1.00").

Override `equals` and compare both `money` and `employeeID`.

Implement a static method `addMoney(IPerson person, int amount)`. This method should add amount to the `money` of the given person.

Hint: Calling `payMoney` with a negative value will pay the person. However, if person is an employee, only half will get added. You can check if a person is an `Employee` with: `person instanceof Employee`. In this case, you should double the value to make sure the correct amount is paid.

Write a getter for `employeeID`.

Exercise 4

Implement the following code in `main` to test your code.

```
IPerson person = new Employee(12345);
Employee.addMoney(person, 200);
System.out.println(((Employee)person).getEmployeeID());
System.out.println(((Person)person).getMoney());
person.payMoney(300);
System.out.println(((Person)person).getMoney());
```