**COS125 Inheritance Example**

A fundamental mechanism for code reuse is **inheritance**, which allows the creation of new types with extended or modified properties of the original type. Inheritance models the ***is-a*** relationship.

Consider a company with two types of workers, hourly and salaried. All workers have a name and an identification number and receive a weekly paycheck. However, the salary computation differs. Hourly workers record number of hours and hourly rate and receive overtime over 40 hours. Salaried workers have a base salary and receive the same amount each week.

Start by identifying data members common to all employees:

* first name
* last name
* identification number

Identify any methods common to all employees:

* toString
* salary

**Worker** will form the super (base) class.

Method **toString** can be written in the base class with the intent it will be **overwritten** in any derived classes.

Method **salary** cannot be written in the base class. However, **salary** is a vital method that must be computable for any kind of worker. This decision tells you **salary** should be an **abstract method** and **Worker** will be an **abstract class**, used as the parent for other classes that model the different types of workers in the company.

Next consider the types of workers which will be described in derived classes. What data members will they add? How will **toString** and **salary** methods be written?

**public abstract class Worker**

**{**

**protected String fName; // first name**

**protected String lName; // last name**

**protected String identification; // identification number**

**public Worker ( ) // POST: empty worker**

**{ fName = "";**

**lName = "";**

**identification = "";**

**}**

**public Worker (String fn, String ln, String id) // POST: worker initialized by parameters**

**{ fName = fn;**

**lName = ln;**

**identification = id;**

**}**

**// accessors and mutators are not listed for brevity**

**…**

**public String toString ( ) // POST: return name and identification**

**{ return fName + " " + lName + " " + identification;**

**}**

**public abstract double salary( ); // POST: return salary**

**}**

**public class HourlyWorker extends Worker**

**{**

**private int hours; // hours worked**

**private double payRate; // hourly pay rate**

**public HourlyWorker ( ) { // POST: empty hourly worker**

**super ();**

**hours = 0;**

**payRate = 0.0;**

**}**

**public HourlyWorker (String fn, String ln, String id, int hr, double pr )**

**// POST: hourly worker constructed from parameters**

**{ super (fn,ln,id);**

**hours = hr;**

**payRate = pr;**

**}**

**// accessors and mutators are not listed for brevity**

**…**

**public String toString ( ) // POST: return all object data**

**{**

**NumberFormat fmt = NumberFormat.getCurrencyInstance();**

**return super.toString( ) + " " + hours + " " + fmt.format(payRate);**

**}**

**public double salary ( ) // POST: return weekly pay including overtime**

**{ if (hours <= 40)**

**return hours \* payRate;**

**else**

**return 40\*payRate + (hours-40)\* payRate\*1.5;**

**}**

**}**

**public class SalariedWorker extends Worker**

**{**

**private double baseSalary; // weekly base salary**

**public SalariedWorker() // POST: empty salaried worker**

**{ super();**

**baseSalary = 0.0;**

**}**

**public SalariedWorker(String fn, String ln, String id, double bs)**

**// POST: salaried worker constructed from parameters**

**{ super(fn,ln,id);**

**baseSalary = bs;**

**}**

**// accessor and mutator are not listed for brevity**

**…**

**public String toString ( ) // POST: return all object data**

**{ NumberFormat fmt = NumberFormat.getCurrencyInstance();**

**return super.toString( ) + " " + fmt.format(baseSalary);**

**}**

**public double salary ( ) // POST: return weekly pay of base**

**{ return baseSalary;**

**}**

**}**

**public class PaycheckProcessor {**

**public static void main(String[] args) throws IOException**

**{**

**String code; // worker code: H for Hourly, S for Salaried**

**String first; // first name**

**String last; // last name**

**String id; // identification number**

**Scanner scan = new Scanner (new File ("payroll.txt"));**

**NumberFormat fmt = NumberFormat.getCurrencyInstance();**

**while (scan.hasNext()) // read all common worker data**

**{ code = scan.next();**

**first = scan.next();**

**last = scan.next();**

**id = scan.next();**

**if (code.equals("H")) // process hourly worker**

**{**

**int hour = scan.nextInt(); // hours worked in week**

**double rate = scan.nextDouble(); // hourly pay rate**

**HourlyWorker hw = new HourlyWorker (first, last, id, hour, rate);**

**System.out.println (hw.toString());**

**System.out.println ("Salary is " + fmt.format(hw.salary()));**

**}**

**else // process salaried worker**

**{**

**double base = scan.nextDouble(); // weekly base salary**

**SalariedWorker sw = new SalariedWorker (first, last, id, base);**

**System.out.println (sw.toString());**

**System.out.println ("Salary is " + fmt.format(sw.salary()));**

**}**

**System.out.println();**

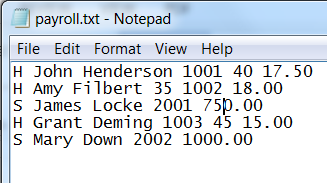
**}**

**scan.close();**

**}**

**}**

Input File:



Console Output:

