# Lab 1

Q1 (Encapsulation):

(a) Create a class **Employee1** that has

**Attributes:**

**name**(String) and **salary**(int)

**Methods**

**getName**, **setName**, **getSalary** and **setSalary**

(b) Complete the following test driver program that creates two employee objects

(**emp1** and **emp2**) and

1. Set the name and salary of **emp1** to "**Chan Tai Man**" and **12000**.

2. Set the name and salary of **emp2** to "**Tam Ping Shing**" and **13500**.

3. Print the details of **emp1** and **emp2**.

4. Increase the salary of "**Chan Tai Man**" by **10%** and the salary of "**Tam Ping Shing**" by **5%**.

5. Print the details of **emp1** and **emp2**.

/

/

Q2 (Inheritance).

Consider the class **Food** class below.

public class **Food** {

protected String name;

protected int price;

public Food() {

name = null;

price = 0;

}

public Food(String name, int price) {

this.name = name;

setPrice(price);

}

public String toString() {

return "name=" + name + ", price=" + price;

}

public void setPrice(int price) {

if (price >= 0)

this.price = price;

else

this.price = 0;

}

}

i) Create a class **Drink** extends class **Food**, and contains

a) a **protected** instance variable **volume (int)**,

b) a constructor that initializes **name**, **price** and **volume**,

c) a method **toString()** to return a string with the **name**, **price** and **volume**,

d) the **setPrice** method to set the price to 5 if price < 5, otherwise call the

super class's **setPrice()** method.

(*Note: the minimum price of a drink is $5*)

The constructor/methods **must** use the constructor/methods of its superclass whenever possible.

public class **Drink** extends Food{

protected int volume;

public Drink(String name, int price, int volume) {

. . . . .

}

public String toString() {

. . . . .

}

public void setPrice(int price) {

. . . . .

}

}

ii) Create a class **Coffee** extends class **Drink**, and contains

a) a **protected** instance variable **isSweet (boolean)**,

b) a constructor that initializes **name**, **price**, **volume** and **isSweet**,

c) a method **toString()** to return a string with the **name**, **price**, **volume** and **isSweet**,

d) the **setPrice** method to set the price to 10 if price < 10, otherwise call the

**setPrice()** method of its superclass.

(*Note: the minimum price of a cup of coffee is $10*)

The constructor/methods **must** use the constructor/methods of its superclass

whenever possible.

A test program **TestFood** and output are given below.

**/**

**/**

Q3 (Polymorphism).

Change the class **Food** in the previous lab to an **abstract class** and its **setPrice()** method to **abstract**. You have to rewrite the **setPrice()** of the class **Drink** as it cannot call the **super.setPrice()** any more. Modify it so that it changes price directly.

Test your classes with the following drive program:

public class TestFood {

public static void main( String s[] ) {

Food f[] = new Food[4];

f[0] = new Drink( "Pepsi", 7, 250 );

f[1] = new Coffee( "Cappuccino", 13, 200, true );

f[2] = new Drink( "Orange Juice", -10, 180 );

f[3] = new Coffee( "Ireland", -11, 200, false );

for ( int i = 0; i < f.length; i++ ) {

System.out.println( "Food " + i + ": " + f[i] );

}

}

}

**Sample Output**

Food 0: name=Pepsi, price=7, volume=250

Food 1: name=Cappuccino, price=13, volume=200, isSweet=true

Food 2: name=Orange Juice, price= 5, volume=180

Food 3: name=Ireland, price= 10, volume=200, isSweet=false

Explain why the following line cannot be compiled.

Food noodle = new Food( "Noodle", 23 );

Q4.

Consider the following **class** and **interface** definitions.

|  |  |
| --- | --- |
| public **abstract** class Staff {  String name;  int id, salary;  char grade;  public Staff (String name,  int id, char grade) {  this.name = name;  this.id = id;  this.grade = grade;  }  public **abstract** void display();  } | **interface** Salary {  public int salary\_A = 4000,  salary\_B = 3000,  salary\_C = 2000,  salary\_other = 1000;  public int computeSalary();  } |

Complete the class **ParttimeStaff** which extends **Staff** and implements interface **Salary** below to

(a) Provide a constructor to accept **name, id, grade, workinghour** and compute the **salary** by calling **computeSalary()**.

(b) Implement the **display()** to print the **name, id, grade, workinghour** and **salary**.

public class **ParttimeStaff** **extends Staff** **implements Salary** {

int workinghour;

public ParttimeStaff(String name, int id, char grade, int workinghour) { . . . . . }

public void display() { . . . . . }

public int computeSalary() {

switch(grade){

case 'A': return salary\_A;

case 'B': return salary\_B;

case 'C': return salary\_C;

default: return salary\_other;

}

}

}

A test program **TestStaff** is given below.

public class **TestStaff** {

public static void main(String args[]) {

ParttimeStaff p1 = new ParttimeStaff(“John”, 123, ‘B’, 20);

ParttimeStaff p2 = new ParttimeStaff(“Mary”, 124, ‘A’, 22);

p1.display();

p2.display();

}

}

**Sample Output**

Name: John, Staff ID: 123, Grade: B, Salary: $3000, Working Hour: 20

Name: Mary, Staff ID: 124, Grade: A, Salary: $4000, Working Hour: 22