Quiz: Inheritance

According to Horstmann and Cornell, the hallmark of inheritance in Java is a relationship between objects that can be characterized as:

a) “Is a”
b) “Is similar to”
c) “Has part of”
d) “Has a common interface with”.
e) “Has the same fields as”
Java Inheritance

If class C inherits class P then
- class C inherits all of the methods and fields of class P.
- class C may contain additional fields and methods not in class P.
- class C may override methods or fields from class P.

In Java the syntax for this is: `C extends P`

In Java, a superclass is also called a
1. Base class, or
2. Parent class

In Java, a subclass is also called a
1. Derived class, or
2. Child class

Meaning of General vs Special

- In common English, special means different from general.
- In science, mathematics and engineering, general means applies to a larger set than does special.

Sometimes, this means general includes special:
- The General Theory of Relativity completely includes the Special Theory of Relativity.
- A general algorithm for finding the shortest path in a graph can be applied to any special cases.

Sometimes, this means the general is the intersection of properties held by each of the special cases:
- In Java, a superclass is more general than its subclass.
- In Java, a subclass, is special case of a superclass.
Inheritance: example from page 174

Manager sylvain = new Manager();
sylvain.setSalary(120000.0);
sylvain.setBonus(50000.0);
System.out.println(sylvain.getSalary());

class Employee
{
    private double salary;
    public void setSalary(double salary)
    {
        this.salary = salary;
    }
    public double getSalary()
    {
        return salary;
    }
}

class Manager extends Employee
{
    private double bonus = 0;
    public void setBonus(double amount)
    {
        bonus = amount;
    }
    public double getSalary()
    {
        return getSalary() + bonus;
    }
}

What the intended control flow?

class Employee
{
    private double salary;
    public void setSalary(double salary)
    {
        this.salary = salary;
    }
    public double getSalary()
    {
        return salary;
    }
}

class Manager extends Employee
{
    private double bonus = 0;
    public void setBonus(double amount)
    {
        bonus = amount;
    }
    public double getSalary()
    {
        return getSalary() + bonus;
    }
}

What is the error?

The field Employee.salary is not visible.

Inheritance: getSalary()

Manager sylvain = new Manager();
sylvain.setSalary(120000.0);
sylvain.setBonus(50000.0);
System.out.println(sylvain.getSalary());

class Employee
{
    private double salary;
    public void setSalary(double salary)
    {
        this.salary = salary;
    }
    public double getSalary()
    {
        return salary;
    }
}

class Manager extends Employee
{
    private double bonus = 0;
    public void setBonus(double amount)
    {
        bonus = amount;
    }
    public double getSalary()
    {
        return getSalary() + bonus;
    }
}

The direct use of salary has been replaced with a call to the accessor method: getSalary()

This compiles, but will not work.

Why?

This is an Infinite loop. Must be: super.getSalary();
final – Find the Two Compile Errors

```java
class Employee
{
    private double salary;
    private final int startYear;
    private final int startMonth;
    public Employee(int year, int month)
    { startYear = year;
    }
    public double getSalary()
    { return salary;
    }
    public void setSalary(double salary)
    { this.salary = salary;
    }
    public void setMonth(int month)
    { startMonth = month;
    }
    public String getStartDate()
    { return startMonth + "/" + startYear;
    }
}
```

The blank final field `startMonth` may not have been initialized.

final field `startMonth` cannot be assigned.

final – Fixed

```java
class Employee
{
    private double salary;
    private final int startYear;
    private final int startMonth;
    public Employee(int year, int month)
    { startYear = year;
        startMonth = month;
    }
    public double getSalary()
    { return salary;
    }
    public void setSalary(double salary)
    { this.salary = salary;
    }
    public String getStartDate()
    { return startMonth + "/" + startYear;
    }
}
```
Adding Error Checking

```java
public Employee(double salary,
       int year, int month)
{
    this.salary = salary;
    if (year < 2000 || year > 2010)
        throw new IllegalArgumentException("bad year");

    if (month < 1 || month > 12)
        throw new IllegalArgumentException("bad month");

    startYear = year;
    startMonth = month;
}
```

Inheritance: Constructor (page 175)

```java
Employee don = new Employee(100000, 2003, 10);
Manager seth = new Manager (70000, 5, 2009, 11);

class Employee
{
    private double salary;
    private final int startYear;
    private final int startMonth;
    public Employee(double salary, int year, int month)
    {
        this.salary = salary;
        if (year < 2000 || year > 2010)
            throw new IllegalArgumentException("bad year");

        if (month < 1 || month > 12)
            throw new IllegalArgumentException("bad month");

        startYear = year;
        startMonth = month;
    }

    public double getSalary()
    {
        return salary;
    }

    public String getStartDate()
    {
        return startMonth + ":" + startYear;
    }
}

class Manager extends Employee
{
    private double bonus;
    public Manager(double salary, double bonus, int year, int m)
    {
        super(salary, year, m);
        this.bonus = bonus;
    }

    public double getSalary()
    {
        return super.getSalary() + bonus;
    }
}
```
Quiz: on Friday

- Polymorphism
- Dynamic Binding
- Casting
- Pascal’s Triangle