ELO330: Programación de sistemas Segundo Certamen 5/11/2003

1.- Responda brevemente:

a) ¿A qué se le llama ***bytecode*** y quién lo genera?

Llamamos bytecode al lenguaje intermedio usado por Java el cual es interpretado por la maquina virtual Java. Lo genera el compilador Java (javac)

b) ¿Cómo se configura la variable CLASSPATH en Linux y cuál es su uso?

Basta con poner:
export CLASSPATH=dir1:dir2:.
en el archivo .bashrc o activarlo en el shell donde se está trabajando. Es usado para indicar los directorios que contienen las clases compiladas requeridas para la compilación o ejecución de un programa Java.

c) ¿Cuáles son las tres ediciones que ofrece Java y cuál es la orientación de cada una de ellas?

Java Enterprise Edition: Orientada al desarrollo de servicios de grandes empresas, Java Standard Edition: Orientada a desarrollo de aplicaciones de propósito general en Java, Java Micro Edition: Orientada a aplicaciones para arquitecturas de hardware con menos recursos como los dispositivos portátiles personales.

d) ¿A qué se le llama compilación just-in-time?

La Compilación just-in-time corresponde a la efectuada en secuencia con la ejecución del código. En el caso de Java, corresponde a compilar el bytecode en código nativo y luego ejecutarlo en la medida que éste es requerido. Código compilado es almacenado durante la ejecución para aprovecharlo en el caso de nuevos accesos al mismo código durante al ejecución.

****

2.- a) ¿Cuáles son tres elementos que caracterizan a un objeto? Cuando se declara:
Employee empleado; ¿Qué elementos se están definiendo?

Su nombre, Su estado, y su comportamiento.
En Employee empleado; se está definiendo un nombre (empleado) y un comportamiento. Nota: incluye comportamiento porque se dice que satisfacer todos los llamados definidos para la clase Employee. Si bien podría referenciar a una instancia de una subclase de Empleado, los llamados permitidos serán sólo aquellos de la clase Employee. Cómo logra cumplir con un determinado comportamiento es otra cosa y depende de la implementación de éste, la cual puede variar de una clase a la subclase debido al ligado dinámico.

b) Suponga que existe la clase Employee con un constructor Employee(String nombre); y las definiciones:

```
Employee a=new Employee("Luis");
Employee b=new Employee("Luis");
```

¿Qué valor lógico tiene la siguiente expresión? Explique.

```
if (a==b) {.....  }
```

*Falso. Esto porque **a** y **b** son dos nombres que representan a distintos objetos. Ambos objetos coinciden en su estado, pero son dos objetos. En Java **a** sería igual a **b** si ambos fueran nombres del mismo objeto.*

c) ¿Cómo se llama el archivo que contiene el programa?

```
public class FirstSample
{  public static void main(String[ ] args)
   {  System.out.println("We will not use 'Hello, elo330!'");
   }
}
```

Muestre los comandos para: Compilar, y luego ejecutar la aplicación este programa.

El archivo debe llamarse FirstSample.java.
Para compilarlo: javac FirstSample.java
Para ejecutarlo: java FirstSample

d) Menciones dos ocasiones en que usted usa la palabra reservada this.

Cuando deseo invocar a un constructor desde otro constructor.
Cuando deseo tener alcance a una variable que se ha hecho inalcanzable por su coincidencia con el nombre de un parámetro de un método o una variable local.

****

3.- Haga un programa en Java que solicite el ingreso de dos números enteros y muestre el resultados de la suma.

```
import javax.swing.*;

public class suma
{
  public static void main(String[] args)
  {
    int a,b;

    String numero_s = JOptionPane.showInputDialog("Ingrese su numero:");
    a = Integer.parseInt(numero_s);
    numero_s = JOptionPane.showInputDialog("Ingrese su segundo numero:");
    b = Integer.parseInt(numero_s);
    System.out.println("La suma es: " + a +" + " + b + " = " + (a+b));
    JOptionPane.showMessageDialog(null, a +" + " + b + " = " + (a+b));
    System.exit(0);
  }
}
```

****

4.- Se desea hacer un programa genérico para calcular el promedio de los elementos de un arreglo.

a) Proponga un código para la interfaz Escalar la cual contiene la operación toDouble(). Ésta retorna un double que representa al objeto.

```
public interface Escalar
{
  double toDouble();
}
```

b) Proponga un método estático llamando ***promedio*** en la clase P4 que retorna el promedio de un arreglo de objetos cada uno de los cuales responden a la interfaz Escalar.

```
public class p4
{
  public static double promedio(Object [ ] a)
  {
    double sum=0;
    for( int i = 0; i < a.length; i++)
      sum+=((Escalar)a[i]).toDouble();
    return (sum/a.length);
  }
}
```

c) Escriba el código necesario para completar alguna de las clases Employee vistas en clases de modo que implemente la interfaz Escalar.

```
import java.util.*;

class Employee implements Escalar
{
  public Employee(String n, double s,
                  int year, int month, int day)
  {
    name = n;
    salary = s;
    GregorianCalendar calendar
      = new GregorianCalendar(year, month - 1, day);
    hireDay = calendar.getTime();
  }

  public double toDouble()
  {
    return salary;
  }

  public String getName()
  {
    return name;
  }

  public double getSalary()
  {
    return salary;
  }

  public Date getHireDay()
  {
    return hireDay;
  }
  public void raiseSalary(double byPercent)
  {
    double raise = salary * byPercent / 100;
    salary += raise;
  }
  private final String name;
  private double salary;
  private Date hireDay;
}
```

****

5.- Para el programa que se acompaña:

a) Indique si el programa tiene errores de compilación. Si los hay, ¿Cómo los corregiría?.

Hay varios errores de compilación:

1.- En el main se invoca al constructor Person() el cual no ha sido definido en la clase Person.

2.- La clase Person debería ser abstracta debido que uno de sus métodos lo es.

3.- Si la clase Person es abstracta habría otro error en el main al tratar de crear una instancia de tal clase.

4.- La variable p del main, sólo tiene validez dentro del cuerpo del for. No puede ser invocada fuera.

Para solucionar estos problemas hay varias formas. Dependiendo de la lógica del programa, en main se podría crear una instancia de Employee o Student en lugar de una instancia de Person. Adicionalmente habría que definir la clase Person como abstract.
Alternativamente, se podría implementar un constructor para Person, y luego implementar el método getDescription en Person, el cual ya no sería abstract.
En ambos casos, habría que cambiar p posterior del for por other.

b) Si la **`LINEA MARCADA`**, se cambia por

```
public String getDescription() { return("Hola\n");}  /*LINEA nueva b*/
```

¿Cuál sería la salida? Explique.

Implementando además el constructor Person() por ejemplo como:

```
Person () {
        name ="vacio";
}
```

Se tiene como salida:

```
Harry Hacker, an employee with a salary of 50000.0
Maria Morris, a student majoring in computer science
vacio, Hola
```

Esto porque por ligado dinámico en el caso de people[0] y people[1], el método getDescription es quel correspondiente al objeto invocado en cada caso, y para el caso de other es el de Person.

c) Si por el contrario la línea marcada se cambia por:

```
public final String getDescription(){ return("Hola\n");}  /*LINEA nueva c */
```

¿Cuál sería la salida? Explique.

El programa no compila porque se intenta sobremontar en método definido como final.

```
import java.text.*;
public class PersonTest
{
```

```
   public static void main(String[] args)
   {
      Person[] people = new Person[2];
      people[0] = new Employee("Harry Hacker", 50000);
      people[1] = new Student("Maria Morris", "computer science");
      Person other = new Person();
      for (int i = 0; i < people.length; i++)
      {
         Person p = people[i];
         System.out.println(p.getName() + ", " + p.getDescription());
      }
      System.out.println(p.getName() + ", " + other.getDescription());
   }
}

class Person
{
   public Person(String n)
   {   name = n;  }
   public abstract String getDescription();  /*LINEA MARCADA */
   public String getName()
   {  return name;
   }
   private String name;
}

class Employee extends Person
{
   public Employee(String n, double s)
   {
      super(n);
      salary = s;
   }
   public String getDescription()
   {        return "an employee with a salary of " + salary;
   }
private double salary;
}

class Student extends Person
{
   public Student(String n, String m)
   {
      super(n);
      major = m;
   }
   public String getDescription()
   {
      return "a student majoring in " + major;
   }
   private String major;
}
```