Programación II

Carla TARAMASCO

Profesora e Investigadora

DECOM & CNRS

mail : alumnos_uv@yahoo.cl

21 mars 2013

Plan de la sección Objetos

- Objetos
  - Crear nuevos Objetos
  - Llamar a Metodos
  - Biblioteca de clase Java
  - Ejercicio de Objetos

Crear objetos

Para crear instancias de clases se debe usar el operador *new* seguido de parentesis. Crea una nueva instancia de la clase y se le asigna memoria. Para esto se llama al metodo constructor de la clase.

- String str = new String()
- Random r = new Random ()
- Auto a = new Auto()

Acceso a las variables de instancia y de clase

Rappel !

Las variables de instancia y de clase se comportan de la misma forma que las locales pero para acceder a ellas se debe usar un punto. Así al lado izquierdo de la notación queda el objeto y al lado derecho la variable.

- myVar.val ;
- a.marca ;

Rappel 2 !

Acceso a las variables de instancia y de clase

Rappel !

Las variables de instancia y de clase se comportan de la misma forma que las locales pero para acceder a ellas se debe usar un punto. Así al lado izquierdo de la notación queda el objeto y al lado derecho la variable.

- myVar.val ;
- a.marca ;

Rappel 2 !

Para cambiar el valor a una variable :

- myVar.val = 80 ;
- a.marca = ”Ford” ;

Ejercicios Iniciales

- Escriba una programa que nos muestre la fecha de hoy. Para eso use la clase Date del paquete java.util (*import java.util.Date ;*)
- Escriba un programa que declare 3 notas y calcule el promedio de estas.
- Escriba una programa que calcule el area de un rectángulo, declarando 2 variables enteras largo y ancho. Si uno de los dos lados es inferior o igual a 0 el programa deberá mostrar un mensaje de ”error”.
- Escriba una programa que compare dos enteros a y b y señale si a es menor, mayor o igual a b

Ejercicio resueltos

```
import java.util.Date;

public class Fecha{
        public static void main (String args[]){
        Date d1;
        d1=new Date();
        // estas lineas puedo reemplazar por:
        //Date d1 = new Date();
        System.out.println("la fecha de hoy es :"
                                              +d1);
        }}
```

Ejercicios resueltos

```
public class Notas2{
        public static void main (String arg[]){

                int nota1=4;
                int nota2=3;
                float nota3=4.6f;
                float promedio;

                promedio = (nota1+nota2+nota3)/3;
                System.out.println("El promedio es"
                                + promedio);

        }}
```

Ejercicios resueltos

```
public class Notas{

        int nota1;
        int nota2;
        int nota3;
        float promedio;

        void promedio(){
                promedio = (nota1+nota2+nota3)/3;
                System.out.println("El promedio es "+ promedio);

        }
        public static void main (String args[]){
                Notas a = new Notas();
                a.nota1 = 5;
                a.nota2= 4;
                a.nota3= 2;
                System.out.println("llamando a promedio");
                a.promedio();
        }
}
```

Ejercicios resueltos

```
public class Area{

        int largo;
        int ancho;
        float areat;

        void calculoArea(){
                if ((largo<=0)||(ancho<=0)){
                        System.out.println("El calculo del area es imposible");}
                else{areat = (largo*ancho);
                        System.out.println("El area es "+ areat);
                }
        }
                public static void main (String args[]){
                        Area a = new Area();
                        a.largo = 15;
                        a.ancho=4;
                        System.out.println("llamando al metodo calcular area");
                        a.calculoArea();
                }
        }
```

Ejercicios resueltos

```
public class Comparar{

        int a;
        int b ;
        float areat;

        void compararEnteros(){
                System.out.println("a es = "+a + "  b es = " +b);
                if (a>b){
                        System.out.println("a es mayor que b");}
                else{
                        if (a<b) {
                                System.out.println("a es menor que b");}
                        else {
                                System.out.println("a es igual a b");
                                }
                }}
                public static void main (String args[]){
                Comparar z = new Comparar();
                z.a = 5;
                z.b=46;
                System.out.println("llamando al metodo comparar");
                z.compararEnteros();
                }
        }
```

Llamar a Metodos

Para llamar a metodos en los objetos es similar al acceso de variables, tambien usa el punto. El objeto que llama al metodo esta en el lado izquierdo y el metodo con sus argumentos al lado derecho

- myObj.method(arg1,arg2) ;
- a.mostrarAtr() ;

Llamar a Metodos

Para llamar a metodos en los objetos es similar al acceso de variables, tambien usa el punto. El objeto que llama al metodo esta en el lado izquierdo y el metodo con sus argumentos al lado derecho

- myObj.method(arg1,arg2) ;
- a.mostrarAtr() ;

Algunos métodos de la clase String :

- length() : nos da el largo de la cadena
- charAt(?) : el carácter que esta en la posición ” ?”
- substring(? ?, ? ?) : el substring entre la posición ? ? y ? ?
- indexOf(’ ?’) : el indice del caracter ’ ?’ o del inicio de la cadena ” ? ? ? ? ?”
- toUpperCase() : la cadena en mayúscula

Ejercicio resuelto String

```
public class TestString{
        public static void main (String args[]){
                String str="Prueba clase String";
                System.out.println("La cadena es "+ str);
                System.out.println("El largo de la cadena es: "+ str.length());
                System.out.println("El caracter en la posicion 5 es: "+ str.charAt(5));
                System.out.println("La sub cadena entre 4 y 7 es: "+ str.substring(4,7));
                System.out.println("El indice de b es: "+ str.indexOf('b'));
                System.out.println("El indice del inicio de la cadena clase es "
                                                + str.indexOf("clase"));
                System.out.println("La cadena en mayuscula: "+ str.toUpperCase());
        }}
```

Ejercicio resuelto de String

```
public class TestString2{
        String str;

        void metodosString(){

                System.out.println("La cadena es "+ str);
                System.out.println("El largo de la cadena es: "+ str.length());
                System.out.println("El caracter en la posicion 5 es: "+ str.charAt(5));
                System.out.println("La sub cadena entre 4 y 7 es: "+ str.substring(4,7));
                System.out.println("El indice de b es: "+ str.indexOf('b'));
                System.out.println("El indice del inicio de la cadena clase es "+
                                                        str.indexOf("clase"));
                System.out.println("La cadena en mayuscula : "+ str.toUpperCase());
        }

        public static void main (String args[]){
                TestString2 mio = new TestString2();
                mio.str="Prueba clase String";
                mio.metodosString();
                        }}
```

Referencias a objetos

Que pasa con pt2 después de cambiar las variables de instancia de pt1 ?

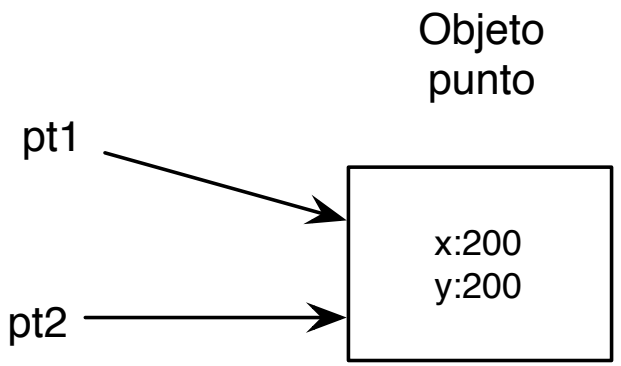
```
import java.awt.Point;

public class TestReferencias{
        public static void main (String args[]){
                Point pt1, pt2;
                pt1 = new Point(100,100);
                pt2=pt1;
                pt1.x=200;
                pt1.y=200;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2.x+","+pt2.y);
                pt1.x=100;
                pt1.y=100;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2.x+","+pt2.y);
                pt2.x=150;
                pt2.y=150;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2.x+","+pt2.y);
                pt2=null;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2);
        }}
```

Referencias a objetos

Cuando se asigna el valor pt1 a pt2, se crea una referencia de pt2 al mismo objeto al cual se refiere pt1. Si cambia el objeto al que se refiere pt2 tambien modificará el objeto al que apunta pt1, ya que ambos se refieren al mismo objeto. Es decir, se crea un objeto punto que referencia a 2 variables (pt1 y pt2).

Referencias a objetos



Referencias a objetos

Que pasa con pt2 y b después de cambiar las variables pt1 y a ?

```
public class TestReferencias1{

        public static void main (String args[]){
                String str1, str2;
                int a=9,b=0;
                str1="mi string test";
                str2=str1;
                b=a;
                System.out.println("Cadena 1= "+str1);
                System.out.println("Cadena 2= "+str2);
                System.out.println("Valor a= "+a);
                System.out.println("Valor b= "+b);
                a=250;
                str1="cambio punto 1 para test ";
                System.out.println("Cadena 1= "+str1);
                System.out.println("Cadena 2= "+str2);
                System.out.println("Valor a= "+a);
                System.out.println("Valor b= "+b);
                b=150;
                str2="otro punto";//str2=new String();
                System.out.println("Cadena 1= "+str1);
                System.out.println("Cadena 2= "+str2);
                System.out.println("Valor a= "+a);
                System.out.println("Valor b= "+b);}}
```

Rappel : Solo los tipos primitivos y String se manejan por valor, es decir, al realizar str2=str1 o b=a se genera una referencia a un mismo objeto pero esta referencia se rompe y se generan 2 objetos al cambiar el valor de alguna de las variables (str2="otro punto") o crear una nueva variable (str2=new String().

Referencias a objetos

Que pasa con pt2 después de cambiar las variables de instancia de pt1 ?

```
public class TestReferencias2{
        int x;
        int y;
        String z;

        public static void main (String args[]){
                TestReferencias2 pt1,pt2;
                pt1 =new TestReferencias2();
                pt1.x=50;
                pt1.y=50;
                pt1.z="60";
                pt2=pt1;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y+","+pt1.z);
                System.out.println("Punto 2= "+pt2.x+","+pt2.y+","+pt2.z);
                pt1.x=100;
                pt1.y=100;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2.x+","+pt2.y);
                pt2.x=350;
                pt2.y=150;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2.x+","+pt2.y);
                pt2=null;
                System.out.println("Punto 1= "+pt1.x+","+pt1.y);
                System.out.println("Punto 2= "+pt2);
        }}
```

Conversión de objetos y tipos primitivos

Para cambiar un valor de un tipo a otro se usa el mecanismo de conversión llamado ”forzar”. El resultado es una nueva referencia o valor por lo cual no afecta al objeto o valor original. Las reglas de conversion dicen relación con los tipos de datos de java. java posee tipo primitivos (int, float, boolean) y tipos objeto (String, Point, Window), por lo cual hay 3 formas de conversion :

- Forzar entre tipos primitivos : de int a float a boolean.
- Forzar entre tipos de objeto : de una instancia de una clase a una instancia de otro clase.
- Convertir tipos primitivos a objetos y después extraer el valor de dichos objetos.

Forzar tipos primitivos

- Los valores booleanos no pueden forzarse a otro tipo.
- Si forzamos a un tipo mas grande que el valor original puede tratarse de manera automática dado que no perderá información.
  Ejemplo : int i = 5 ;
  double d = i ;
- Para convertir de un valor de tipo grande a uno pequeño se debe usar el forzado explicito :
  - (tipo de dato) valor
  - (int) x
  - (int) (x/y)

  Rappel : la precedencia del forzado es mas alta que la aritmética

Forzar Objetos

- Al forzar un objeto el tipo de dato no cambia, sólo cambia la manera en que el compilador va a tratar a dicho objeto.
- Solo se puede forzar instancia de una clase a otra si estas están relacionadas por la herencia. Se puede forzar un objeto solo a cierta distancia de la sub o super clase de la clase a la que pertenece, no a cualquier clase.
- Se pueden usar instancias de la subclase en cualquier superclase (un objeto de la subclase es también un objeto de la superclase por lo cual puede ser tratado como instancia de la superclase).
- Forzar un objeto a una superclase de ese objeto (se perderá la información proporcionada por la subclase) se debe usar un forzado explícito :
  - (nombre clase) objeto
  - a1=(Moto) a ;

Forzar Objetos

Habitualmente los métodos son declarados para ser genéricos, posiblemente devolviendo o aceptando un tipo Object. Si se necesita acceder a un parámetro por un tipo especifico puede forzarse.

```
public class TestConvertir{

        public void myMethod(Object param) {
                Number num = (Number)param;
                System.out.println("El valor es: " + num);
                }
        public static void main (String args[]){
                TestConvertir a = new TestConvertir();
                a.myMethod(67.8f);
        }}
```

Forzar Objetos

Para verificar si una referencia a un objeto es una instancia de cierta clase o de su padre se usa el operador : **instanceof**. Este operador tiene un objeto a la izquierda y el nombre de una clase a derecha. La expresion regresa **true** o **false** dependiendo si el objeto es una instancia de la clase nombrada o de alguna de sus subclases.

```
public class TestConvertir2{

                public void miMetodo(Object param) {
                        if (param instanceof Number) {
                                Number num = (Number)param;
                                System.out.println("Objeto tipo Number : " + num);
                        }
                }

                public static void main (String args[]){
                        TestConvertir2 a = new TestConvertir2();
                        a.miMetodo(22.0f);
                }}
```

Forzar tipos primitivos a Objetos

No es posible forzar de tipos de datos primitivos a objetos ni vice versa. Sin embargo el paquete java.lang incluye varias clases especiales que corresponden a tipos de datos primitivos.
Integer para **ints**, **Float** para **floats**, **Boolean** para **boolenos**, etc.
Para utilizar los métodos de estas clases puede crear objetos equivalentes a todos los tipos de datos primitivos mediante el uso de **new**.
Integer intObjeto = new Integer(23) ;
Esto crea una instancia de la clase **Integer** con el valor 23, este valor puede ser tratado como objeto.

Forzar Objetos a tipo de dato primitivos

Existen metodos para regresar los valores a tipos primitivos. Por ejemplo : **intValues()**, que extrae un tipo primitivo int de un objeto Integer.
int elEntero = intObjeto.intValues() ;

Ejemplo resumen : conversión de tipo de datos (objeto y tipos primitivos)

```
//import java.lang.Integer;
public class TestConvertir3{

                public void miMetodo(Object param) {
                        if (param instanceof Number) {
                                Number num = (Number)param;
                                System.out.println("Objeto tipo Number : " + num);
                                double d = num.doubleValue();
                                System.out.println("Tipo primitivo double : " + d);
                        }
                }

                public static void main (String args[]){
                        TestConvertir3 a = new TestConvertir3();
                        Integer intObjeto = new Integer((int)(23.4));
                        System.out.println("Objeto tipo Integer: " + intObjeto);
                        a.miMetodo(intObjeto);
                }}
```

Comparar Objetos

La mayoría de los operadores funciona solo en tipos primitivos no en objetos excepto los operadores de igualdad **==** y **!=**.
Estos operadores prueban si los dos operandos se refieren al mismo objeto, es decir, los operadores de igualdad no evaluarán si los valores de dos objetos son iguales sino mas bien, si referencian al mismo objeto.

```
public class TestIgual{

        public static void main (String args[]){
                String str1, str2;
                str1="test operador de igualdad en objetos";
                str2=str1;
                System.out.println("Cadena 1 :"+ str1);
                System.out.println("Cadena 2 :"+ str2);
                System.out.println("Son el mismo objeto? :"+ (str1==str2));

                str2=new String(str1); //str2="objeto 2"; con cualquiera de estas
                //instrucciones se rompe la referencia creada con str2=str1
                //generando 2 objetos de tipo String : str1 y str2.
                System.out.println("Cadena 1 :"+ str1);
                System.out.println("Cadena 2 :"+ str2);
                System.out.println("Son el mismo objeto? :"+ (str1==str2));
                System.out.println("Tienen el mismo valor? :"+ str1.equals(str2));

        }}
```

Determinar la clase de un objeto

String nombre = obj.getClass().getName() ;

El método *getClass()* da como resultado un objeto *Class* que posee un método llamado *getName()* que regresa la cadena de caracteres con el nombre de la clase.

Algunos paquetes de Java

Determinar la clase de un objeto

String nombre = obj.getClass().getName() ;

El método *getClass()* da como resultado un objeto *Class* que posee un método llamado *getName()* que regresa la cadena de caracteres con el nombre de la clase.

Algunos paquetes de Java

Cada biblioteca de java ofrece un conjunto de clases disponible.

- *java.lang :* clases que se aplican al lenguaje mismo entre ellas la clase *Object, System,String*. También contiene las clases especiales para tipos primitivos *Integer, Float, Character*, etc.
- *java.util :* clases utilitarias como *Date* y clases de colección de datos como *Vector*.
- *java.io :* clases de entrada y salida, para escribir y leer flujos de datos y manejar archivos.
- *java.net :* clases para soporte de red, clases como *URL*.
- *java.awt :* clases para trabajar con una interfaz gráfica de usuario y procesar imagenes, incluye las clases *Window,Menu, Button,Font, Image,* entre otras.