

# Clases y Objetos en Java

ELO329: Diseño y Programación Orientados a  
Objetos

# Construyendo clases nuevas

- La forma más simple de una clase en Java es:

```
Class Nombre_de_la_Clase {  
    /* constructores */  
    /* métodos */  
    /* atributos */  
}
```

- Ver ejemplo Employee.java y EmployeeTest.java
- Podemos definir más de una clase por archivo, pero desde fuera del archivo sólo tendremos acceso a aquella con nombre igual que el archivo.
- Clases de nombre distinto al archivo sólo son accesibles dentro del archivo.

# Constructores en Java

- Son métodos con algunas peculiaridades:
  - Tienen igual nombre que la clase
  - Normalmente tienen parámetros pero es ok que no los tenga.
  - Son invocados principalmente con new
  - Pueden ser invocados con this desde otro constructor.
  - No tiene tipo retornado
  - No tienen return explícito
  - Java provee constructor por defecto (); es decir, sin parámetros, cuando ningún otro constructor ha sido creado.
  - Podemos proveer uno o más constructores. Esto es un tipo de sobrecarga de métodos (igual nombre con distintos parámetros)
  - El compilador busca el constructor usando “firma” nombre constructor + lista de parámetros

# Constructores en Java

- Inicializa objetos nuevos siguiendo el siguiente orden:
  - 1. Localiza memoria
  - 2. Asigna valores a variables (0, 0.0, null, ...)
  - 3. Según el orden de aparición en la clase se ejecutan las inicializaciones allí hechas
  - 4. Llama constructor de Superclase.
  - 5. Se ejecutan las sentencias del constructor.
- La primera sentencia puede ser:
  - `super( ... )` para llamar a un constructor de la clase base (o padre o superclase)
  - `this( ... )` para invocar a otro de la misma clase
- Ver Ejemplo ConstructorTest.java

# Inicialización de Campos (datos)

- Podemos proveer el valor inicial de un dato en su declaración.

Como en:

```
int a = 20;
```

- Si esta asignación requiere más lógica (cómputo), usamos el bloque de inicialización. Como en:

```
{  
    a=20; // luego que a ha sido declarado.  
}
```

Este código se ejecuta antes de cualquier constructor definido.

- En caso de campos estáticos, precedemos el bloque con la palabra reservada static. Ejemplo:

```
static {  
    INCHES_PER_CM = 2.54;  
}
```

Este código se ejecuta antes de la primera instrucción del main.

- Ejemplo: ConstructorTest.java

# Creación de objetos nuevos

- Se usa el constructor de la clase  
`MiClase a = new MiClase();`
- Todos los objetos son creados en el heap (memoria asignada dinámicamente durante la ejecución).
- Lo que se retorna es una referencia al nuevo objeto (puede ser pensada como un puntero).
- Java tiene un proceso de recolección de basura (Garbage Collection) que automáticamente recupera zonas no referenciadas.
- Si deseamos hacer algún tipo de limpieza antes de liberar el espacio de un objeto, la clase debería incluir un método con nombre **finalize()**. Éste se invoca justo antes de recolectar su memoria.

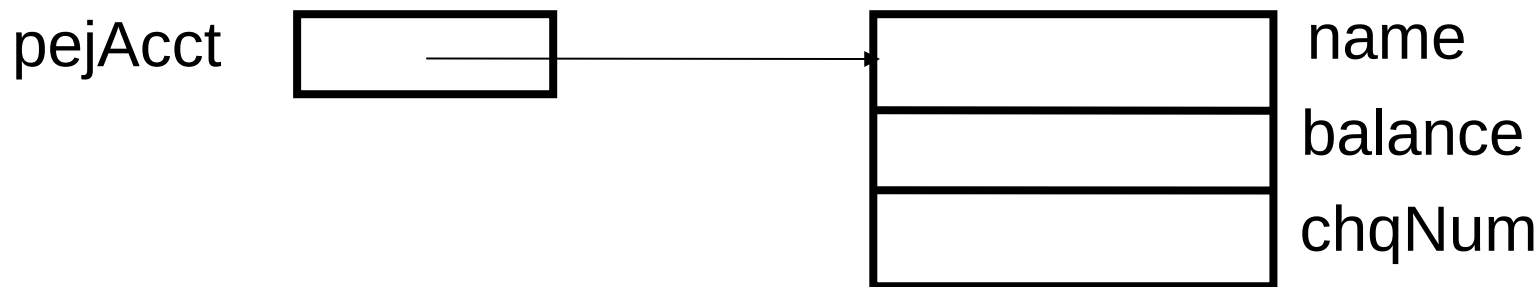
# Identificadores de Objetos v/s los objetos

Cheque pejAcct;

pejAcct  // Referencia nula

`pejAcct.deposit(1000000);` // error

`pejAcct = new Cheque("Peter", 1000, 40);`



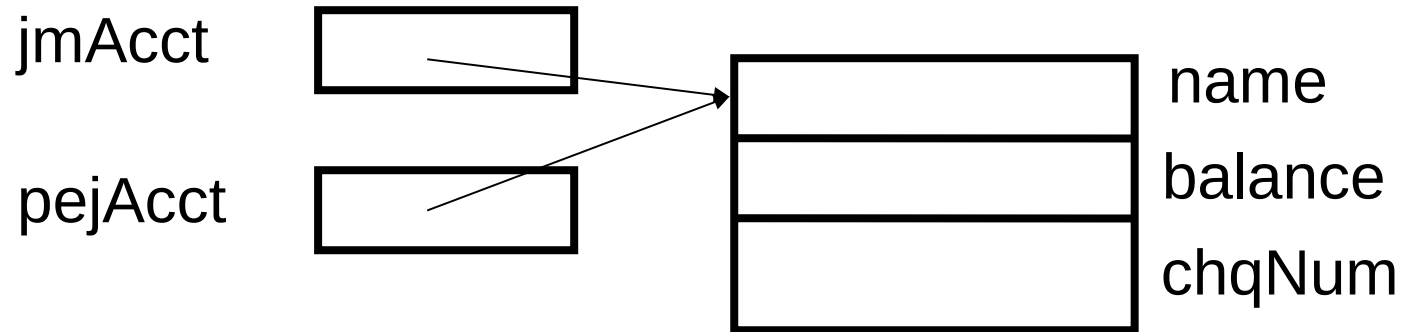
Este ejemplo asume que la clase Cheque ya existe y posee miembros datos (= atributos): name, balance y chqNum

# Asignación de objetos

Cheque jmAcct;



jmAcct = pejAcct;





# Referencias

- Los objetos son referenciados
- Esta es una forma “controlada” de usar: Direcciones y punteros
- Al declarar una instancia de una clase obtenemos una referencia a esa instancia.
- Mientras no se asigne un objeto con new, su valor es null.
- En caso de tipos primitivos (8) se tiene la variable y acceso directo (no es referencia)
  - byte, short, int, long, float, double, char, boolean

# Implicancias de referencias

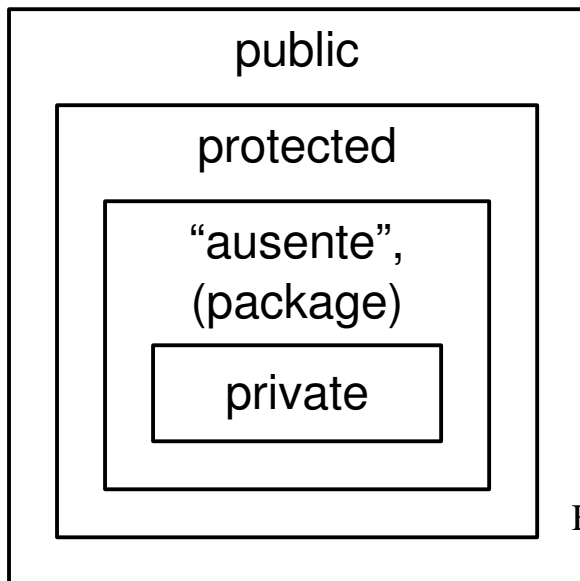
- Los identificadores de objetos son **referencias**
  - referencia significa puntero (i.e. no el contenido)
- = es copiar la referencia
  - Usar método **clone** para crear copia del objeto completo (más adelante).
- == es comparación de referencias
  - Usar **equals** para comparar contenidos
- obj.aMethod(pejAcct) pasa un referencia
- obj.aMethod(tipo\_básico) pasa el valor
- return pejAcct retorna una referencia
  - Si queremos retornar una copia, usar clone para crearla y luego retornarla

# Visibilidad de clases, métodos y atributos

- Para crear buenas abstracciones debemos dejar visible (accesible) al usuario de una clase sólo aquello que es estrictamente necesario.
- Para esto Java posee varios **modificadores del nivel de acceso** (o visibilidad). Éstos preceden los nombres de clases, método o atributos.
- Estos **modificadores** son: público, protegido, paquete, y privado.

# Visibilidad de clases, métodos y atributos

Modificador de Acceso	Visibilidad
<b>private</b>	<b>Sólo visible en la clase</b>
<b>Sin modificador (omitido)</b>	<b>En clase y el paquete</b>
<b>protected</b>	<b>En clase su-clases y paquete</b>
<b>public</b>	<b>Desde todas partes</b>



Modificador	Clase	Package	Subclass	World
private	si	no	no	no
Sin modificador	si	si	no	no
protected	si	si	si	no
public	si	si	si	si

¿Cómo sabe el compilador y la jvm  
dónde están las clases de una  
aplicación?

Para ubicar localización de ejecutables vimos  
necesidad de configurar la variable PATH.

Para clases, configurar variable CLASSPATH.

# Compilación (re-visitado)

- El compilador busca la definición de cada clase que encuentra en el archivo <nombreDeClase>.java
- Para señalar al compilador dónde buscar debemos configurar la variable de ambiente: **CLASSPATH**
  - El compilador y la JVM buscan los archivos en el directorio actual.
  - Si el proyecto está compuesto por varias clases en diferentes directorios, javac y java buscan las clases en los directorios indicados en la variable de ambiente CLASSPATH.
- Si .class tiene **fecha más antigua que .java**, javac re-compila el archivo .java.
- Se destaca así la **importancia de los nombres de archivo** de las clases que deseamos visibles desde otros archivos.
- En Linux esta variable se configura con
  - export CLASSPATH=/home/user/classdir1:/home/user/classdir2:.
- El Windows también se debe configurar la variable de ambiente.