UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Objetos y Clases en Java

ELO-329: Diseño y Programación Orientados a Objetos

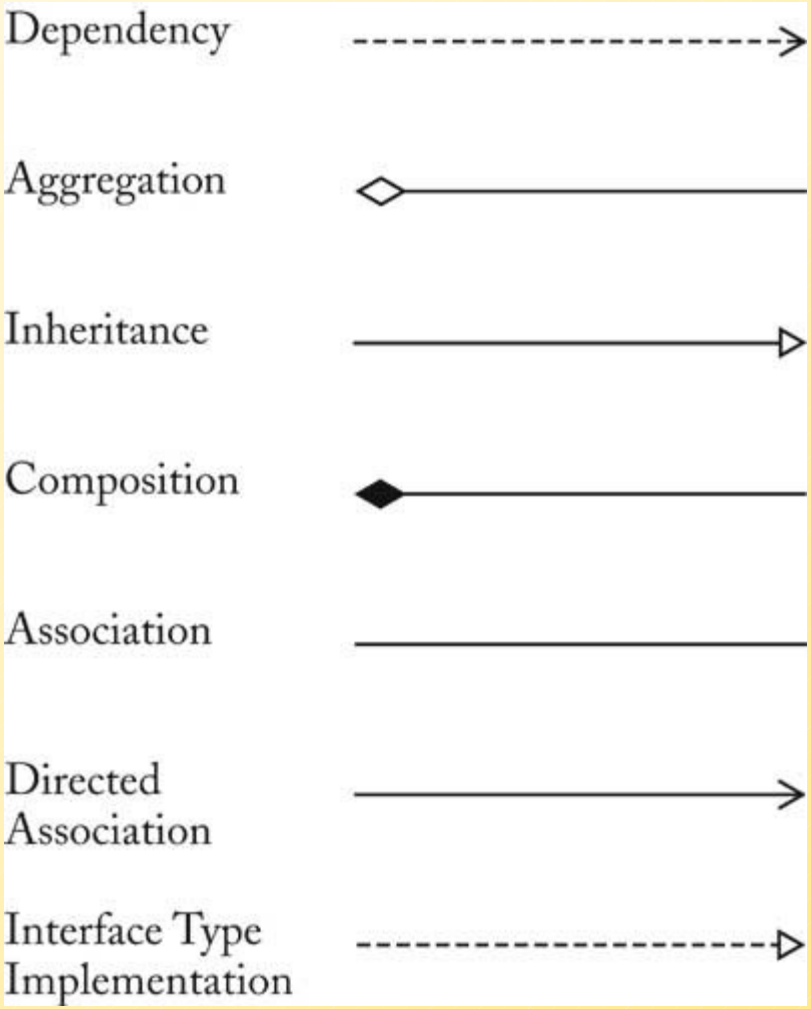
Elementos de Análisis y Diseño orientado a objetos

- Notación y relaciones entre clases:
- Las clases son abstracciones para los entes o cosas que constituyen el sub-mundo o modelo de la realidad donde existe el sistema bajo construcción.
- Las clases se relacionan entre si de varias formas.
- Existe una notación para expresar gráficamente las clases de un sistema y la relación entre ellas.

Relación entre Clases

- A usa B envía mensajes a B
- A tiene-un B A contiene atributo clase B
  También llamada Agregación 
- A es-un B herencia de B 
- Diagrama de clases muestras estas asociaciones.
  Por ejemplo Jgrasp puede generar estos diagramas a partir del código Java. Otras editores también.
  Ej: Rational Rose y Together (pagadas)
  ArgoUML, GebtleWare Open source.

Relación entre clases



Relación entre clases

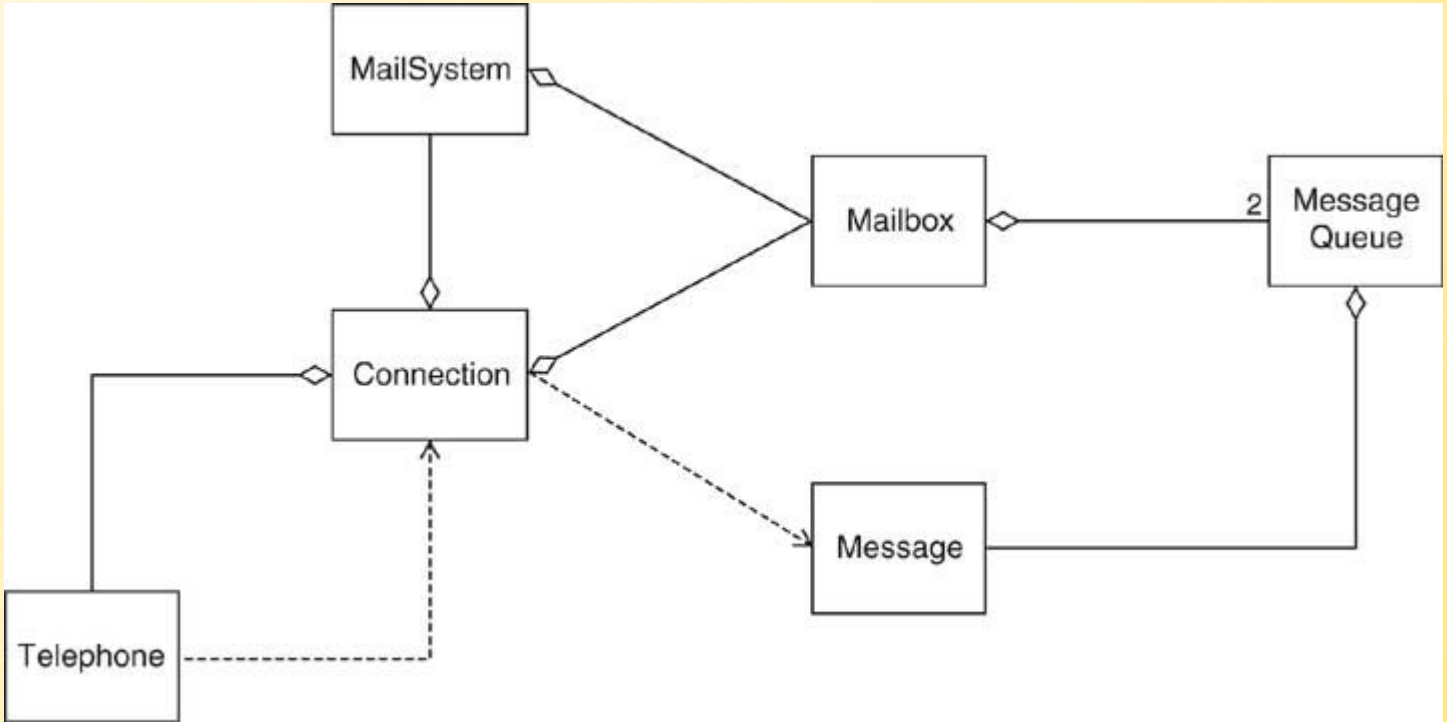
- Agregación: Una cola de mensajes tiene 0 ó más mensajes.
- * cualquier número (0 ó mas)
- 1..* Uno o más
- 0..1 Cero o uno
- 1 Exactamente uno

Relación entre clases

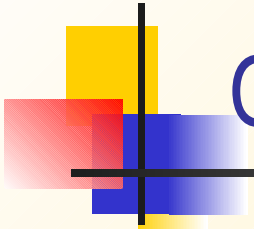
- Composición: Caso especial de agregación
- El objeto contenido no existe fuera de la clase
- La cola de mensajes de este ejemplo sólo está contenida en el mail box.

Diagrama de Clases

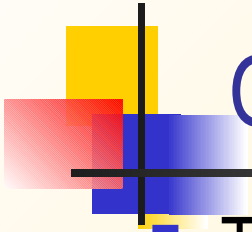
- Diagrama que muestra las relaciones entre las clases de un sistema.
- Usa notación UML (Unified Modeling Language).



Creación de objetos nuevos

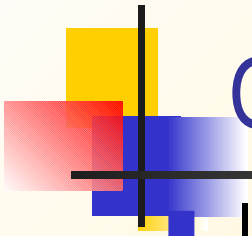


- Se usa el constructor de la clase
  MiClase a = new MiClase();
- Todos los objetos son creados en el heap (memoria asignada dinámicamente durante la ejecución).
- Lo que se retorna es una referencia al nuevo objeto (puede ser pensada como puntero).
- Nota — no existe destructor (en C++ sí)
  Java tiene un proceso de recolección de basura (Garbage Collection) que automáticamente recupera zonas no referenciadas.



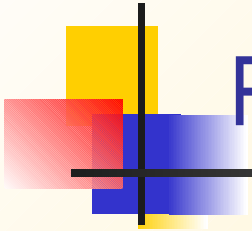
Constructores

- Tiene igual nombre que la clase
- Pueden tener parámetros
- Son invocados principalmente con **new**
- No tiene tipo retornado
- No **return** explícito
- Java provee constructor por defecto **()**
- 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

- Inicializa objetos nuevos:
  - 1. Localiza memoria
  - 2. Asigna valores por defecto a variables (0, 0.0, null, …)
  - 3. Llama constructor de Superclase (más adelante)
  - 4. Sentencias restantes son ejecutadas
- La primera sentencia puede ser:
  - super(…) para llamar al constructor de la clase base (o padre o superclase)
  - this(…) invoca a otro constructor



Referencias

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

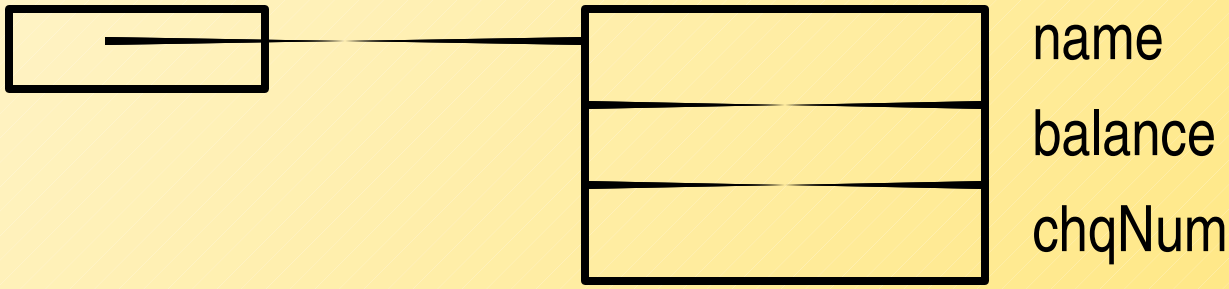
Definiendo variables

Cheque pejAcct;

pejAcct  Referencia nula

pejAcct.deposit(1000000); // error

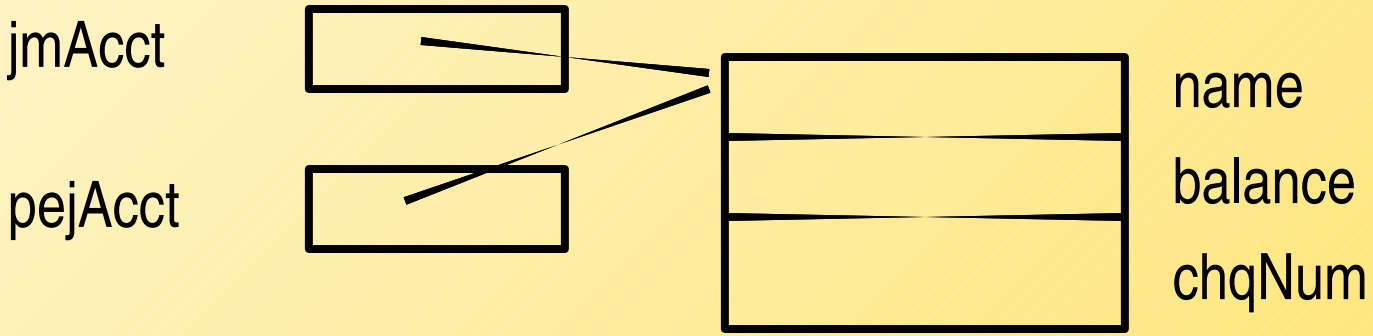
pejAcct = new Cheque("Peter", 1000, 40);



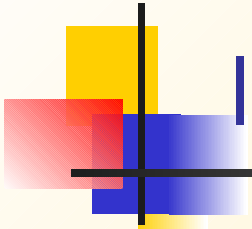
Asignación

Cheque jmAcct;

jmAcct = pejAcct;



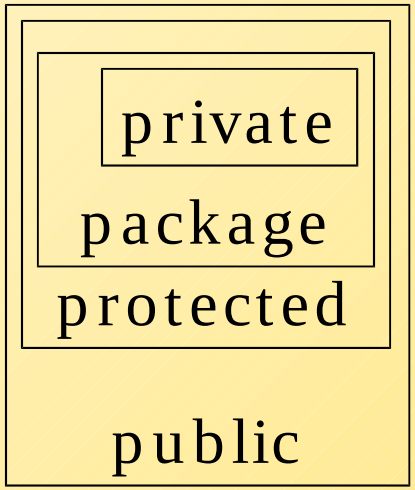
Implicancias de referencias



- La identidad 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.
- == es comparación de referencias
  - Usar **equals** para comparar contenidos
- java usa paso de parámetros por valor, “Call by value”.
- aMethod(pejAcct) pasa un referencia
- aMethod(tipo_básico) pasa el valor
- return pejAcct retorna una referencia
  - Usar clone para crear una copia, y luego retornarla

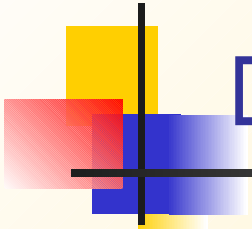
Control de acceso

Modificador de acceso	*Visibilidad*
public	Todas partes
protected	en sub-classes & pkg
“omitido”	En el paquete
private	Sólo en la clase



Documentación

- Para la clase ponerla inmediatamente antes de la clase y ser encerrado entre /** y */
- Para los métodos: usar los rótulos
  - @param *variable* descripción
  - @return descripción
  - @throws descripción de clase
- Para los datos públicos: /** ...*/
- Comentarios Generales:
  - @author nombre
  - @version texto
  - @since texto
  - @see link
  - Ejemplo: @see cl.utfsm.elo.Employee#raiseSalary(double)



Documentación

- Se pueden usar todo tipo de rótulos html incrustados.
- ¿Cómo generar la documentación?:
  - javadoc -d docDirectory *.java
- Para la documentación de un paquete:
  - javadoc -d docDirectory nameOfPackage
- Ejemplo:
  - Account.java
  - index.html generado con javadoc -d AccountDoc *.java

Rutas para clases

- Primero incluir la ruta del compilador y máquina virtual java en la variable PATH.
- Luego la ruta para la búsqueda de todas las clases: CLASSPATH
  - El compilador y el interprete java 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.
- En Linux ELO ésta se configura con
  - export CLASSPATH=/home/user/classdir1: /home/user/classdir2:.
- El Windows también se debe fijar la variable de ambiente.