Programación OO con Java

- Interfaz
  - Una interfaz es una colección de definiciones de métodos (sin implementaciones) y de valores constantes.
  - Una clase puede implementar más de una interfaz.
  - Las interfaces son lo más parecido a la herencia múltiple (pero no lo es).

Programación OO con Java

- Ejemplo
  - `public class Auto {`
    - `String modelo,color;`
    - `public void recorrido() {`
      - `System.out.println("Santigo-Valparaiso");`
    - `}`
    - `}`
  - `interface Velocidad {`
    - `public float velocidadActual();`
    - `}`

Programación OO con Java

- interface Aceleracion {
  - public float aceleracionActual();
  - }

  - public class AutoMovil extends Auto implements Velocidad,Aceleracion {
  - public float velocidadActual() {
    - System.out.println("Velocidad = 100");
  - }
    - public float aceleracionActual() {
    - System.out.println("Aceleración = 12");
  - }
  - }

  - ver:Prueba.java,ChaoInterface.java,HelloImpInter.java y HelloInterface.java

Programación OO con Java

- Clases importantes en Java
  - Clase Object
    - boolean o.equals(Object)
    - Object o.clone()
    - Class o.getClass()
    - String o.toString()
  - Clase Character
    - boolean Character.isDigit(char)
    - boolean Character.isLetter(char)
    - boolean Character.isSpace(char)

Programación OO con Java

- Clase String
  - int cad.length()
  - char cad.charAt(int)
  - boolean cad.equals(Object)
  - boolean cad.equalsIgnoreCase(Object)
  - Clase StringBuffer
  - StringBuffer sb1 = new StringBuffer(4);
  - sb1=”Hola”;
  - StringBuffer sb2 = new StringBuffer(“Toma sentado.”);
  - sb2.insert(6,”café ”);
  - System.out.println(sb2.toString());
  - Resultado: Toma café sentado.

Programación OO con Java

- Clase StringTokenizer
  - Permite dividir un string en pedazos(tokens).
  - Ejemplo:
    - `StringTokenizer st = new StringTokenizer("es una prueba"," " );`
    - `while (st.hasMoreTokens()) {`
    - `println(st.nextToken());`
    - `}`
  - Resultado:
    - **es**
      - » **una**
      - » **prueba**

Programación OO con Java

- Ejercicio:

- Cree un programa que:

- - Primero lea el String que separa las palabras o “tokens”
- - Luego lea esas palabras y las imprima en pantalla junto con su largo.
- - Y termine su ejecución al ingresar el usuario “*”.

- ver: Tokens.java

Programación OO con Java

- Clase Random
  - Se utiliza para generar secuencias aleatorias.
  - Random ran = new Random();
    - ran.nextInt(int n);
      - » Retorna un número aleatoria desde cero hasta n
  - Clase Date
  - Clase Properties
  - Clase Runtime
  - Permite ejecutar comandos en procesos separados
  - Ejemplo
    - Process p = Runtime.getRuntime().exec("xterm");
    - ver: comando.java

Programación OO con Java

- Clase Math
  - Es una clase sin ningún constructor, y todos sus métodos son estáticos
  - int Math.abs(int)
  - long Math.abs(long)
  - float Math.abs(float)
  - double Math.abs(double)
  - double Math.random()
  - double Math.PI

Manejo de Errores

- Excepciones
  - Las excepciones son la manera que ofrece Java de manejar los errores en tiempo de ejecución. Muchos lenguajes imperativos, cuando tenían un error de este clase lo que hacían era detener la ejecución del programa. Las excepciones nos permiten escribir código que nos permita manejar ese error y continuar (si lo estimamos conveniente) con la ejecución del programa.

Manejo de Errores

- try…catch...finally
  - try {}
    - Con el bloque try marcamos en que parte del código esperamos que se genere un error.
  - catch{}
    - Con catch atrapamos el error y enviamos algún mensaje si es necesario.
  - finally
    - El código en este bloque se ejecuta siempre, se produzca o no la excepción.

Manejo de Errores

- Un bloque try puede tener asignado más de un bloque catch
- Clase Exception
  - String e.getMessage()
    - Devuelve el mensaje detallado si existe o null en caso contrario.
  - void e.printStackTrace()
    - Escribe por la salida de error estándar una lista que nos permitirá localizar donde se generó el error. Es muy útil para depurar, pero no es conveniente que lo usuarios finales vean estas cosas.

Manejo de Errores

- Ejercicio
  - Construya su propia excepción que notifique en pantalla que fue creada
  - Haga un programa que cree esa excepción
  - Modifique su excepción y ahora implemente el constructor que reciba el mensaje
  - Use el programa para ver los errores que produce
  - Ahora encapsule su código con *try* y *catch*

    - ver: Hello.java y FileIsCorruptedException.java