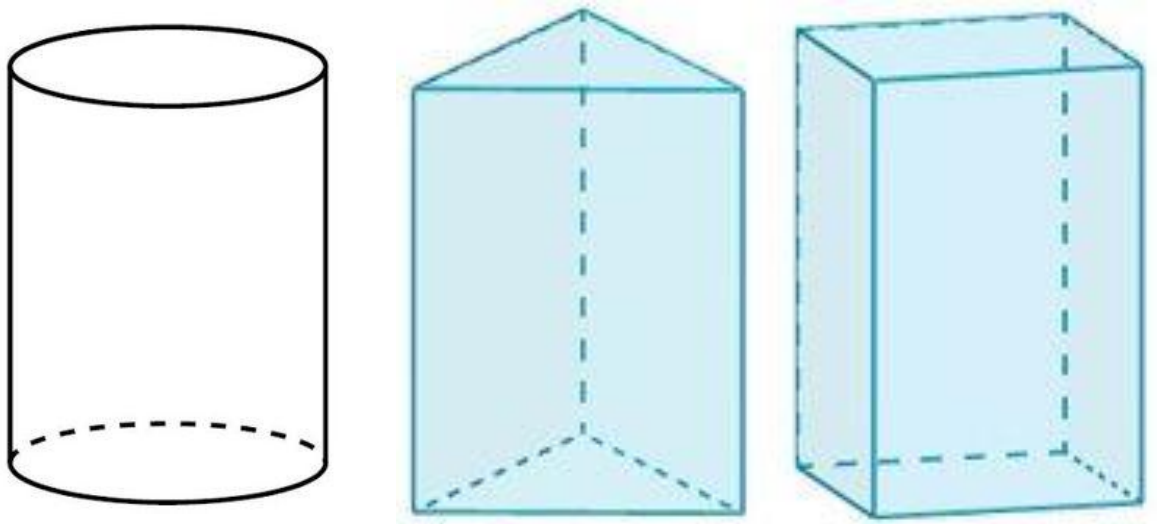
Programación 2

Tecnicatura en Desarrollo de Aplicaciones Informáticas

Figuras 3D

Figura3D



Simplificación en la cual debemos calcular el volumen de la figura

Cilindro-PrismaTriangular-PrismaRectangular

Juego de Dados

Nueva Funcionalidad

---

un jugador es tramposo y utiliza dados cargados que favorecen la salida del **5** y el **6** respectivamente la mitad de las veces que se tira el dado

Dados Cargados

```
public class Dado{
   int valor;
   int caras;

  public int tirar() {
    return (Math.radom()*caras)+1;
}
```

```
public class DadoCargado5 extends Dado{

  public int tirar() {
    if (Math.random()>0.5)
       return super.tirar();
    else
       return 5;
}
```

```
public class DadoCargado6 extends Dado{

  public int tirar() {
    if (Math.random()>0.5)
       return super.tirar();
    else
       return 6;
}
```

Mal uso de la herencia: no voy a crear una clase por cada valor de una variable de instancia que cambie

Dados Cargados

```
public class Dado{
   int valor;
   int caras;

  public int tirar() {
    return (Math.radom()*caras)+1;
}
```

```
public class DadoCargado extends Dado{

  int ladoCargado;
  public int tirar() {
    if (Math.random()>0.5)
       return super.tirar();
    else
       return ladoCargado;
}
```

Dados Cargados

¿Qué otras clases debo modificar?

Como?

Clase Vs Instancia

No puede existir clases iguales

Dos clases iguales son las misma

CUANDO LA DIFERENCIA ES SOLO UNA CONSTANTE, entonces también son la misma clase

Malos ejemplos

PersonaJuan

PersonaPedro ---> “Juan”

----> Existe un nombre, las dos son personas con una variable Nombre

Mal en Dados

Clase padre Dado

DadoCargado5 y DadoCargado6 ---> Los dos son DadoCargado y el valor en realidad es una variable!!!

DESAPARECEN DADOCARGADO5 y DADOCARGADO6

Reconocer fácil el error

NO TIENE QUE HABER CONSTANTES EN EL CÓDIGO

```
return “juan”

return 84;

return “a”;
```

```
if (x>150)

if (nombre.equals(“juan”))
```

```
return “juan”;
reemplazar

return nombreADevolver;
```

La Clase Object

Object

Object es una clase

La clase Object es la superclase de todas las clases

Todas las clases heredan directa o indirectamente de Object

Object

```
public class Object {

      public boolean equals( Object obj );

      public String toString( )

      protected void finalize( )

        ...}
```

Estos métodos pueden ser redefinidos en las subclases si es necesario

equals()

- Retorna true cuando dos objetos tienen iguales valores

- La implementación por defecto compara referencias con ==

equals() Ejemplo Dado

```
public class Dado{
   public boolean equals( Object obj ) { // PARA REDEFINIR
// EL MÉTODO NO PUEDO CAMBIAR LA SIGNATURA DEL MISMO
       Dado other = (Dado) obj; // SI NO ES UN DADO DA ERROR
//MÁS ADELANTE VAMOS A VER COMO SE SOLUCIONA
      return  this.getValor() == other.getVAlor();
    }
}
```

equals() Ejemplo Dado

```
Dado d1 = new Dado(3);

Dado d2 = new Dado(3);

 (d1 == d2);     // returns false

d1.equals(d2);      // returns true
```

toString()

- toString( ) se usa para proveer una representación del objeto como una cadena de caracteres
- Invocada automáticamente cuando utilizamos
  - System.out.println() y “+”
  - String s = “The object is “ + obj;
  - System.out.println(obj);
- La implementación por defecto de la clase Object retorna el nombre de la clase y la ubicación en memoria del objeto

toString() Ejemplo

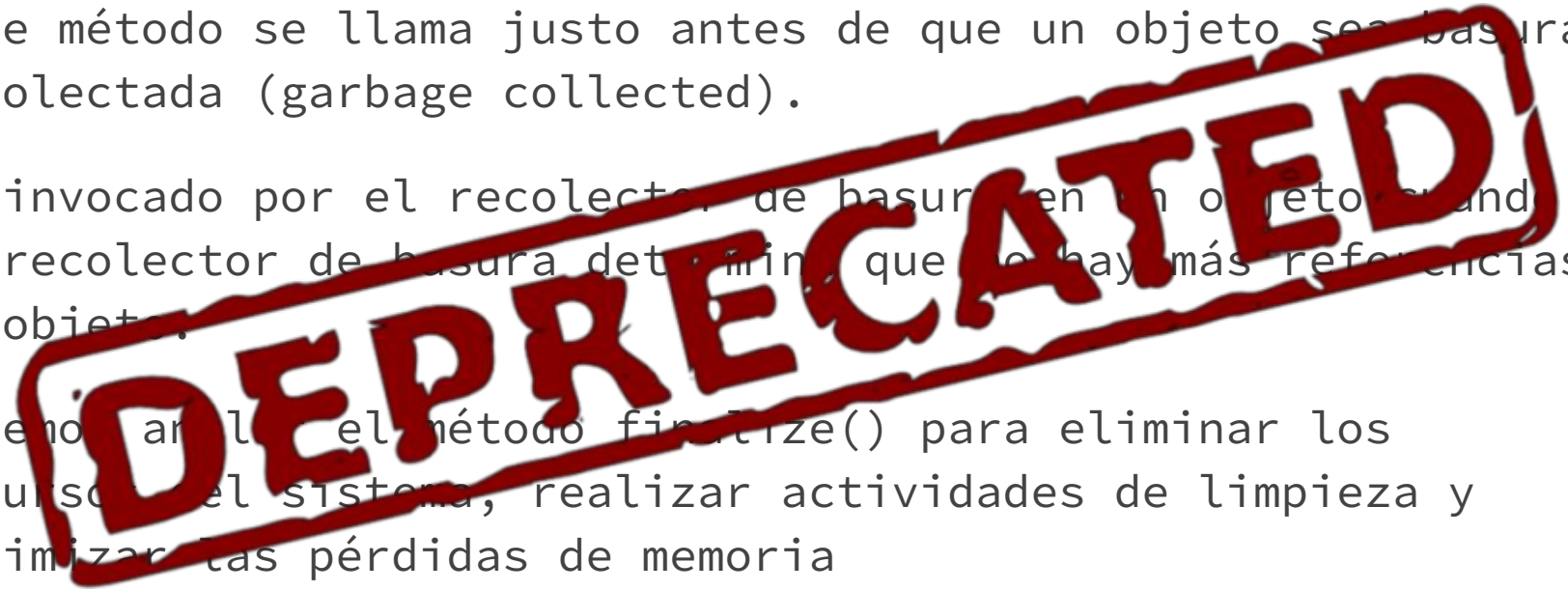
```
public class Dado {


 public String toString( ) {
   return “Dado con el valor = " + this.getValor();
 }
```

finalize()

Este método se llama justo antes de que un objeto sea basura recolectada (garbage collected).

Es invocado por el recolector de basura en un objeto cuando el recolector de basura determina que no hay más referencias al objeto.

Debemos anular el método finalize() para eliminar los recursos del sistema, realizar actividades de limpieza y minimizar las pérdidas de memoria



Colecciones

Límites de los Arreglos

Que problema tenemos con los arreglos?

Colecciones Dinámicas

ArrayList

ArrayList es una clase en Java import **java.util.ArrayList;**

Proporciona una estructura de datos dinámica para almacenar elementos de manera consecutiva.

A diferencia de los arrays estáticos, los ArrayLists pueden crecer o disminuir automáticamente según sea necesario.

ArrayList

```
ArrayList<String> nombres = new ArrayList<>();

nombres.add("Alice");

nombres.add("Bob");

nombres.add("Charlie");
```

ArrayList

```
String primerNombre = nombres.get(0); // Alice

nombres.set(1, "Robert"); //Modifica el valor de la posición 1

nombres.add(1,”Juan”); //agrega un elemento en la posicion 1 y
realiza un corrimiento de los restantes

nombres.remove(2); // Elimina "Robert" (corrimiento)

int tamaño = nombres.size(); // 3

int indice = nombres.indexOf("Alice"); // 0
```

ArrayList

```
for (int i =0; i<nombres.size();i++) {

    System.out.println(nombres.get(i));

}
```