



# Optimización aplicaciones para dispositivos móviles

M.C. Juan Carlos Olivares Rojas

# Optimización del tamaño

- Evitar el uso de objetos siempre que sea posible.
- Cuando usamos objetos, debemos reciclarlos siempre que se pueda.
- Limpiar objetos explícitamente cuando se dejen de usar. Los recolectores de basura no son del todo eficientes.



# Optimización del tamaño

- Usar un ofuscador.
- Realizar empaquetamiento de las clases a través de un archivo jar.
- La optimización especial es muy importante



# Optimización de velocidad

Eliminar evaluaciones innecesarias

```
for(int i=0; i<size(); i++)  
    a = (b+c) / i;
```

- Optimizado:

```
int tmp = b+c;  
int s = size();  
for(int i=0; i<s; i++)  
    a = tmp / i;
```



# Optimización de velocidad

- Eliminar subexpresiones comunes

```
b = Math.abs(a) * c;
```

```
d = e / (Math.abs(a) + b);
```

- Optimizado:

```
int tmp = Math.abs(a);
```

```
b = tmp * c;
```

```
d = e / (tmp + b);
```



# Optimización de velocidad

- Aprovechar las variables locales

```
for (int i=0; i <1000; i++)  
    a = obj.b * i;
```

- Optimizado:

```
int localb = obj.b;  
for (int i=0; i <1000; i++)  
    a = localb * i;
```



# Optimización de velocidad

- Expandir los ciclos
- `for(int i=0; i <1000, i++)`
  - `a[i] = 25;`
- Optimizado:
- `for(int i=0; i <100; i++) {`
  - `a[i++] = 25;`
  - `a[i++] = 25; //8 veces más`
- `}`



# Optimización de velocidad

- Si se usan métodos gráficos solo actualizar las partes de la pantalla que cambian.
- Recolector de basura: evitar la creación de objetos intermedios, cada vez que se crea una nueva cadena se crea un objeto intermedio. En Java en lugar de String se recomienda StringBuffer.



# Referencias

- M. Prieto, “Desarrollo de juegos con J2ME”, Alfaomega Ra-Ma, México, 2005, ISBN: 970-15-1093-3.



# ¿Preguntas?

