Nuevas Tecnologías de la Programación

17 de Febrero de 2011

1. (1 punto) Explica en qué consiste el *buffering* en Xlib y explica una de las condiciones para que se envíe el buffer al servidor.

2. (1 punto) Explica cómo influyen los siguientes elementos del contexto gráfico de X a la hora de dibujar algo con una primitiva gráfica: `clip_mask`, `function`, `foreground`.

3. (2 ptos) Dado el siguiente código Java, escribe una clase que permita guardar una colección de objetos cuya clase pueda ser cualquier subclase de `Figura`. Debe contener al menos lo siguiente:

   - Un método para añadir un nuevo objeto (de cualquier subclase de `Figura`) a la colección.
   - Un método para recuperar el elemento de la posición `i`.
   - Un método para imprimir en la salida estándar el área de todos los elementos de la colección.

```
abstract class Figura {
  double dim1, dim2;
  Figura(double a, double b) {
    dim1 = a;
    dim2 = b;
  }
  abstract double area();
}
class Rectangulo extends Figura {
  Rectangulo(double a, double b) {
    super(a, b);
  }
  double area() {
    return dim1 * dim2;
  }
}
class Triangulo extends Figura {
  Triangulo(double a, double b) {
    super(a, b);
  }
  double area() {
    return dim1 * dim2 / 2;
  }
}
```

4. (2 ptos) En el código Java mostrado más abajo, el interfaz `Selector` sirve para implementar un patrón *iterador* (usado para recorrer los objetos guardados en una colección). Como puede verse, la clase `Sequence` permite guardar una colección de objetos de cualquier clase en su dato miembro `items` (un array de `Object`). La clase `SequenceSelector` representa el iterador concreto para la clase `Sequence`. Su dato miembro `i` guarda la posición del siguiente objeto

a leer de la colección. Completa el código de los métodos `end()`, `current()` y `next()` de la siguiente forma:

- `end()`: Devolverá `true` si hemos llegado al final de la colección de objetos. Devolverá `false` en otro caso.
- `current()`: Devuelve el objeto en la posición actual de la colección.
- `next()`: Mueve la posición actual a la siguiente.

Indica también qué escribe este programa en la salida estándar.

```
interface Selector {
  boolean end();
  Object current();
  void next();
}
public class Sequence {
  private Object[] items;
  private int next = 0;
  public Sequence(int size) { items = new Object[size]; }
  public void add(Object x) {
    if(next < items.length)
      items[next++] = x;
  }
  private class SequenceSelector implements Selector {
    private int i = 0;
    public boolean end() { }
    public Object current() { }
    public void next() { }
  }
  public Selector selector() {
    return new SequenceSelector();
  }
  public static void main(String[] args) {
    Sequence sequence = new Sequence(10);
    for(int i = 0; i < 10; i++)
      sequence.add(Integer.toString(i));
    Selector selector = sequence.selector();
    while(!selector.end()) {
      System.out.print(selector.current() + " ");
      selector.next();
    }
  }
}
```

5. (4 puntos) Escribe un programa en Qt que muestre las coordenadas x e y dónde hemos pinchado con el ratón dentro de un *QWidget*. Tales coordenadas se mostrarán en objetos *QLabel*. El programa debe disponer también de un botón para salir del programa.