UNIVERSIDAD ICESI	**Primer Parcial**	Programación en red 081

Parte Teórica - Valor 1.6

1. Indique cuál afirmación es cierta, respecto a las comunicaciones por Multicast. Para las afirmaciones falsas, indique brevemente porqué las descarta. Selección (0.1) Justificación (0.3)

   a) El nodo de la red que genera el paquete con la información a enviar, crea una copia de acuerdo a la cantidad de miembros del grupo, y los envía a través de la red.
   b) Es posible enviar información mediante Multicast, usando el método send de la clase Socket, aunque el desempeño puede resultar un poco lento.
   c) La entrega de información de los paquetes a través de Multicast, está garantizada a todos los destinatarios del grupo
   d) Todas las anteriores son falsas.

2. (0.4) Señale la o las características que considere, definen al protocolo UDP. Para las opciones no seleccionadas, indique brevemente porqué las descarta. Selección (0.1) Justificación (0.3)

   a) Rapidez
   b) Entrega secuencial de la información
   c) Independencia entre los paquetes que transporta
   d) No orientado a conexión

3. (0.4) Indique cuál o cuáles afirmaciones son falsas, respecto a las direcciones IP. Para las afirmaciones falsas, indique brevemente porqué las seleccionó. Selección (0.1) Justificación (0.3)

   a) Las direcciones IP identifican de manera única los equipos en la red.
   b) Las direcciones IP públicas son aquellas visibles desde cualquier punto en Internet.
   c) En Internet pueden haber dos direcciones IP iguales, siempre y cuando en la comunicación se especifiquen bien los puertos a los cuales se dirige la información.
   d) Existen direcciones IP públicas y privadas. Las direcciones privadas permiten IP's repetidas, siempre y cuando estén en redes distintas.

4. (0.4) Mencione por lo menos dos diferencias entre una Intranet e Internet.

Parte práctica - Valor 3.4

Se requiere una aplicación Cliente/Servidor que le permita a un usuario personalizar la presentación de un texto.

Para escoger la presentación, el servidor pondrá a disposición del usuario un listado con los tipos de letra disponibles, cuatro posibles tamaños y tres posibles colores. El usuario deberá escoger las características del mensaje, y con esto, el servidor creará localmente un texto con el mensaje "Mensaje de prueba", de acuerdo a las especificaciones indicadas por el usuario. Luego de generar el mensaje, el servidor creará un archivo .jpg que almacene la imagen con el texto personalizado, y luego, le enviará dicho archivo al usuario, para que pueda visualizar, en la imagen, cómo quedaría su texto.

El servidor podrá atender a varios usuarios al tiempo, que deseen personalizar sus mensajes.

Nota: Para la manipulación del archivo JPG, puede hacer uso de los métodos save("nombreArchivo.ext"), loadBytes("ArchivoAConvertirEnArregloBytes.ext") y saveBytes("nombreArchivo.ext", arregloDeBytes). Puede asumir que el archivo generado pesa menos de 64KB

5. (2.4) Elabore el análisis de la interacción Cliente/Servidor para la aplicación descrita. Indique claramente qué cosas se ejecutarían en el setup y en el draw.

6. (1.0) Elabore el código del método recibeArchivo( ) de la aplicación **Cliente**, en el cual se recibirá la información para generar la imagen correspondiente al mensaje personalizado por el usuario.

Descripción de métodos

[1]Name save()

Examples
```
line(20, 20, 80, 80);
// Saves a TIFF file named "diagonal.tif"
save("diagonal.tif");
// Saves a TARGA file named "cross.tga"
line(80, 20, 20, 80);
save("cross.tga");
```

Description Saves an image from the display window. Images are saved in TIFF, TARGA, JPEG, and PNG format depending on the extension within the **filename** parameter. For example, "image.tif" will have a TIFF image and "image.png" will save a PNG image. If no extension is included in the filename, the image will save in TIFF format and **.tif** will be added to the name. These files are saved to the sketch's folder, which may be opened by selecting "Show sketch folder" from the "Sketch" menu. It is not possible to use **save()** while running the program in a web browser.

Syntax `save(filename)`

Parameters filename String: any sequence of letters and numbers

Usage Application

[1] Tomado de www.processing.org

[2]Name

saveBytes()

Examples

```
byte[] nums = { 0, 34, 5, 127, 52};

// now write the bytes to a file
saveBytes("numbers.dat", nums);
```

Description

Opposite of **loadBytes()**, will write an entire array of bytes to a file. The data is saved in binary format. This file is saved to the sketch's folder, which is opened by selecting "Show sketch folder" from the "Sketch" menu.

It is not possible to use saveXxxxx() methods inside a web browser unless the sketch is signed. To save a file back to a server, see the save to web example.

Syntax

`saveBytes(filename, bytes)`

Parameters

filename	name of file to write to
bytes	array of bytes to be written

Usage

Application

[2] Tomado de www.processing.org

Name

loadBytes()

Examples

```
// open a file and read its binary data
byte b[] = loadBytes("something.dat");

// print each value, from 0 to 255
for (int i = 0; i < b.length; i++) {
  // every tenth number, start a new line
  if ((i % 10) == 0) {
    println();
  }

  // bytes are from -128 to 127, this converts to 0 to 255
  int a = b[i] & 0xff;
  print(a + " ");
}
// print a blank line at the end
println();
```

Description

Reads the contents of a file or url and places it in a byte array. If a file is specified, it must be located in the sketch's "data" directory/folder.

The filename parameter can also be a URL to a file found online. For security reasons, a Processing sketch found online can only download files from the same server from which it came. Getting around this restriction requires a signed applet.

Syntax

`loadBytes(filename);`

Parameters

filename	String: name of a file in the data folder or a URL.	

Usage

Web & Application