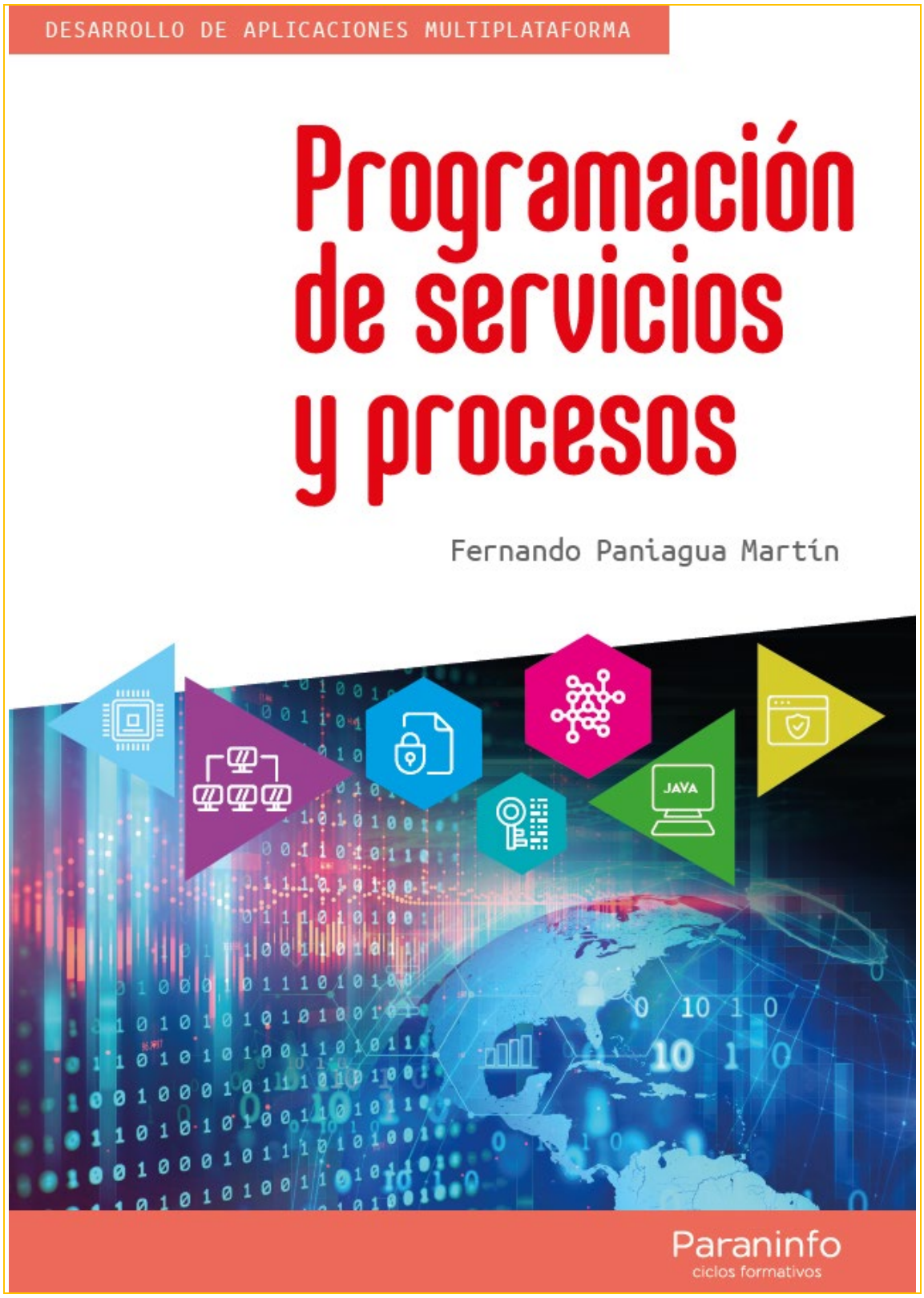

ACTUALIZACIÓN

Página 50. Título de la tabla 2.2.

Pone: Tabla 2.2. Clases del paquete java.lang implicadas en la programación concurrente

Debe poner: Tabla 2.2. Clases del paquete java.lang y java.util implicadas en la programación concurrente

Página 58. Tabla 2.6. Primera fila.

Pone: ConcurrentHashMap Clase Equivalente a un HashMap sincronizado.

Debe poner: ConcurrentHashMap Clase Equivalente a una Hashtable sincronizada.

Página 92. Enunciado de la actividad de comprobación 2.7.

Pone: ¿Qué clase del paquete java.lang representa una tarea programable?

Debe poner: ¿Qué clase del paquete java.util representa una tarea programable?

Página 169. Código fuente.

En la línea 14 falta la letra **o** de **localhost**.

En la línea 15 falta la letra **l** de Calculadora.

```
13              try {
14                      Registry registro = LocateRegistry.getRegistry("lo
    calhost", 5555);
15                      calculadora = (ICalculadora) registro.lookup("Cal
    culadora");
16              } catch (Exception e) {
```

ERRATAS ACTIVIDADES RESUELTAS

SOLUCIÓN ACTIVIDAD RESUELTA 4.2. PÁGINA 148.

La línea 6 de código debe finalizar con un punto.

```
1  public static void main(String[] args) {
2      try {
3              String esquema = "https://";
4              String servidor = "www.imdb.com";
5              String path = "/find";
6              String texto = URLEncoder.
   encode("Tiburón",StandardCharsets.UTF_8.name());
7              String parametros = "?q=" + texto;
8              GestorPeticionesHTTPParametros gp = new
   GestorPeticionesHTTPParametros();
```

SOLUCIÓN ACTIVIDAD RESUELTA 4.2. PÁGINA 148.

En la línea 11 de código pone GestorPeticionesHTTP y debe poner GestorPeticionesHTTPParametros.

```
10              StringBuilder resultado =
    gp.getContenidoMetodoGet(direccion);
11              GestorPeticionesHTTP.writeFile("G:/tiburon_movie.html",
    resultado.toString());
12              System.out.println("Descarga finalizada");
```

SOLUCIÓN ACTIVIDAD RESUELTA 4.3. PÁGINA 150.

La línea 30 de código debe finalizar con un punto.

```
29              HttpResponse<Path> response =
30                      httpClient.send(request, HttpResponse
    BodyHandlers.ofFile(Path.of(path)));
31
```

SOLUCIÓN ACTIVIDAD RESUELTA 4.4. PÁGINA 153.

La línea de código 48 debe finalizar con un punto.

```
45      private boolean subirFichero(String path) throws IOException {
46              File ficheroLocal = new File(path);
47              InputStream is = new FileInputStream(ficheroLocal);
48              boolean enviado = clienteFTP.storeFile(ficheroLocal
    getName(), is);
49              is.close();
```

SOLUCIÓN ACTIVIDAD RESUELTA 4.8. PÁGINA 162.

La línea 40 de código debe finalizar con un punto.

```
39                      String from = direccionesOrigen[0].toString();
40                      Address[] direccionesDestino = mensaje
    getRecipients(RecipientType.TO);
41                      String to = direccionesDestino[0].toString();
```

SOLUCIÓN ACTIVIDAD RESUELTA 4.8. PÁGINA 162.

La línea 43 de código debe finalizar con un paréntesis de cierre.

```
43                      MimeMultipart mimeMultipart = (MimeMultipart
    mensaje.getContent();
```

SOLUCIÓN ACTIVIDAD RESUELTA 4.8. PÁGINA 162.

La línea 45 de código debe finalizar con un paréntesis de cierre.

```
44                      if
    (mimeMultipart.getBodyPart(0).isMimeType("text/plain")) {
45                          String textoMensaje = (String)
    mimeMultipart.getBodyPart(0).getContent();
46                          System.out.printf("De %s A %s Asun
    Mensaje: %s\n", from, to, subject, textoMensaje);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.2. PÁGINA 191.

La línea 19 de código debe finalizar con un punto.

```
    getBytes(ENCODING_TYPE));
19                  Files.write(new File(identificador + ".credencial").
    toPath(), resumen);
20                  mostrarResumenHexadecimal(resumen);
21              } catch (Exception e) {
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 196.

La línea 16 de código debe terminar con una barra /.

```
    Exception {
16              Cipher cipher = Cipher.getInstance("AES/ECB/
    PKCS5Padding");
17              cipher.init(Cipher.ENCRYPT_MODE, key);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 196.

La línea 18 de código debe terminar con un punto.

```
    PKCS5Padding");
17              cipher.init(Cipher.ENCRYPT_MODE, key);
18              byte[] cipherText = cipher.doFinal(textoEnClaro.
    getBytes());
19              return Base64.getEncoder().encodeToString(cipherText);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 197.

La línea 25 de código debe terminar con un punto.

```
25              byte[] plainText = cipher.doFinal(Base64.getDecoder().
    decode(textoCifrado));
26              return new String(plainText);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 197.

La línea 14 de código debe terminar con un punto.

```
fuerte es 3842873110";
13              try {
14                      Key clave = AESSimpleManager.
obtenerClave(PASSWORD, LONGITUD_BLOQUE);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 197.

La línea 16 de código debe terminar con un punto.

```
15                      String textoEnClaro = TEXTO_EN_CLARO;
16                      String textoCifrado = AESSimpleManager.
cifrar(textoEnClaro, clave);
17                      PrintWriter pw = new PrintWriter(NOMBRE_FICHERO);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 198.

La línea 16 de código debe terminar con un punto.

```
14              try {
15                      File file = new File(NOMBRE_FICHERO);
16                      Key clave = AESSimpleManager.
obtenerClave(PASSWORD, LONGITUD_BLOQUE);
17                      BufferedReader br = new BufferedReader(new
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.3. PÁGINA 198.

La línea 19 de código debe terminar con un punto.

```
18                      String textoCifrado = br.readLine();
19                      String textoEnClaro = AESSimpleManager.
descifrar(textoCifrado, clave);
20                      br.close();
21                      System.out.println("El texto descifrado es:" +
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4 PÁGINA 201.

La línea 21 de código debe terminar con un punto.

```
20      public static KeyPair generarClaves() throws NoSuchAlgorithmException {
21              KeyPairGenerator generador = KeyPairGenerator.
getInstance("RSA");
22              generador.initialize(2048);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4. PÁGINA 202.

La línea 38 de código debe terminar con un punto.

```
38              byte[] bytesClavePublica = Files.
readAllBytes(ficheroClavePublica.toPath());
39              KeyFactory keyFactory = KeyFactory.getInstance("RSA");
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4. PÁGINA 202.

La línea 19 de código debe terminar con un punto.

```
    X509EncodedKeySpec(bytesClavePublica);
41              PublicKey clavePublica = keyFactory.
    generatePublic(publicKeySpec);
42              return clavePublica;
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4. PÁGINA 202.

La línea 47 de código debe terminar con un punto.

```
45      public static PrivateKey getClavePrivada() throws Exception {
46              File ficheroClavePrivada = new
    File(FICHERO_CLAVE_PRIVADA);
47              byte[] bytesClavePrivada = Files.
    readAllBytes(ficheroClavePrivada.toPath());
48              KeyFactory keyFactory = KeyFactory.getInstance("RSA");
49              EncodedKeySpec publicKeySpec = new PKCS8EncodedKey
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4. PÁGINA 202.

La línea 50 de código debe terminar con un punto.

```
    Spec(bytesClavePrivada);
50              PrivateKey clavePrivada = keyFactory.
    generatePrivate(publicKeySpec);
51              return clavePrivada;
52      }
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4. PÁGINA 203.

La línea 17 de código debe terminar con un punto.

```
14                  throws Exception {
15              Cipher descifrador = Cipher.getInstance("RSA");
16              descifrador.init(Cipher.DECRYPT_MODE, clave);
17              byte[] mensajeDescifrado = descifrador.
    doFinal(mensajeCifrado);
18              return mensajeDescifrado;
19      }
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.4. PÁGINA 203.

La línea 26 de código debe terminar con un punto.

```
24              try {
25                      clavePublica = ClavesManager.getClavePublica();
26                      byte[] mensajeCifrado = Files.
    readAllBytes(fichero.toPath());
27                      byte[] mensajeDescifrado =
    descifrar(mensajeCifrado, clavePublica);
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.5. PÁGINA 206.

La línea 21 de código debe terminar con un punto.

```
    public static KeyPair generarClaves() throws
    NoSuchAlgorithmException {
21              KeyPairGenerator generador = KeyPairGenerator.
    getInstance("DSA");
22              generador.initialize(512);//Admite 512, 768 ó 1024
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.5. PÁGINA 206.

La línea 38 de código debe terminar con un punto.

```
37              File ficheroClavePublica = new File(FICHERO_CLAVE_PUBLICA);
38              byte[] bytesClavePublica = Files.
    readAllBytes(ficheroClavePublica.toPath());
39              KeyFactory keyFactory = KeyFactory.getInstance("DSA");
```

SOLUCIÓN ACTIVIDAD RESUELTA 5.5. PÁGINA 206.

La línea 41 de código debe terminar con un punto.

```
40              EncodedKeySpec publicKeySpec = new
    X509EncodedKeySpec(bytesClavePublica);
41              PublicKey clavePublica = keyFactory.
    generatePublic(publicKeySpec);
42              return clavePublica;
```