1 Crear el código Java

Primero escribimos la clase java EjemploString

```
package u4.jni00;

public class EjemploString {

    public native String replaceString(String sourceString,
            String strToReplace, String replaceString);

    static {
        System.loadLibrary("BibliotecaString");
    }

    public static void main(String[] args) {
        EjemploString ex = new EjemploString();
        String str1 = "";
        String str2 = "";
        str1 = "Sky Black";
        str2 = ex.replaceString(str1, "Black", "Blue");
        System.out.println("La cadena antes: " + str1);
        System.out.println("La cadena despues: " + str2);
    }
}//end class EjemploString
```

2 Crear el código y las bibliotecas nativas

Para escribir el código en C++, debemos utilizar la herramienta javah (incluida en el JDK) para generar un archivo de cabecera. Este archivo de cabecera contiene los prototipos de las funciones que deben implementarse en C++. En primer lugar se compila el código java y, a continuación se ejecuta esta herramienta con el archivo class. Opcionalmente para compilar la clase EjemploString podemos utilizar la herramienta ant con un archivo build.xml como el siguiente:

```
<project name="JNI_EjemploString" basedir="." default="main">

    <property name="src.dir"     value="newpkgroot"/>

    <property name="build.dir"   value="Build"/>
    <property name="classes.dir" value="${build.dir}/classes"/>
    <property name="imagenes.dir" value="${classes.dir}/Imagenes"/>
    <property name="jar.dir"     value="${build.dir}/jar"/>

    <property name="main-class"  value="u4.jni00.EjemploString"/>

    <target name="clean">
        <delete dir="${build.dir}"/>
    </target>
```

```
    <target name="compile">
        <mkdir dir="${classes.dir}"/>
        <javac srcdir="${src.dir}" destdir="${classes.dir}"/>
<!--
<copy todir="${imagenes.dir}">
<fileset dir="${src.dir}/images"/>
</copy>
-->
    </target>

    <target name="jar" depends="compile">
        <mkdir dir="${jar.dir}"/>
<jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}">
            <manifest>
                <attribute name="Main-Class" value="${main-class}"/>
            </manifest>
        </jar>
    </target>

    <target name="run" depends="jar">
        <java jar="${jar.dir}/${ant.project.name}.jar" fork="true"/>
<!--
cd Build/classes/
java -Djava.library.path=~/BibliotecaString/   u4.jni00.EjemploString
En el directorio ~/BibliotecaString/ debe estar el archivo
libBibliotecaString.so
-->
    </target>

    <target name="clean-build" depends="clean,jar"/>

    <target name="main" depends="clean,run"/>

</project>
```

Las ubicaciones de los archivos build.xml y EjemploString.java antes de compilar usando ant deben ser las siguientes:

```
.
|-- BibliotecaString
|-- build.xml
‘-- newpkgroot
    ‘-- u4
        ‘-- jni00
            ‘-- EjemploString.java
```

Después de ejecutar el comando ant compile debemos tener algo como esto:

```
.
|-- BibliotecaString
|-- Build
|   `-- classes
|       `-- u4
|           `-- jni00
|               `-- EjemploString.class
|-- build.xml
|-- newpkgroot
    `-- u4
        `-- jni00
            `-- EjemploString.java
```

Al ejecutar javah debemos especificar el nombre de la clase (no el nombre del archivo) como primer parámetro.

```
$ mkdir -p BibliotecaString/include/
$ cd Build/classes/
$ javah -d ../../BibliotecaString/include/ u4.jni00.EjemploString
```

Después de ejecutar estos comandos debemos tener

```
.
|-- BibliotecaString
|   `-- include
|       `-- u4_jni00_EjemploString.h
|-- Build
|   `-- classes
|       `-- u4
|           `-- jni00
|               `-- EjemploString.class
|-- build.xml
|-- newpkgroot
    `-- u4
        `-- jni00
            `-- EjemploString.java
```

Ahora mostramos una posible implementación de la biblioteca BibliotecaString: (libBibliotecaString.so en linux o BibliotecaString.dll en Windows).

```
/**
 * bibliotecastring.c Archivo de implementacion de la biblioteca
 * BibliotecaString
 */
#include "u4_jni00_EjemploString.h"

JNIEXPORT jstring JNICALL Java_u4_jni00_EjemploString_replaceString
  (JNIEnv *env, jobject obj,
  jstring _srcString, jstring _strToReplace, jstring _replString){
  const char *searchStr, *findStr, *replStr, *found;
```

```
  jstring newStr=NULL;
  int index;

  searchStr=env->GetStringUTFChars(_srcString, NULL);
  findStr=env->GetStringUTFChars(_strToReplace, NULL);
  replStr=env->GetStringUTFChars(_replString, NULL);

  found=strstr(searchStr, findStr);

  if(found!=NULL){
    char *newStringTemp;
    index=found-searchStr;
    newStringTemp=new char[strlen(searchStr)+strlen(replStr)+1];
    strcpy(newStringTemp, searchStr);
    newStringTemp[index]=0;
    strcat(newStringTemp, replStr);
    strcat(newStringTemp, &searchStr[index+strlen(findStr)]);
    newString=env->NewStringUTF((const char*)newStringTemp);
  }
  env->ReleaseStringUTFChars(_srcString, searchStr);
  env->ReleaseStringUTFChars(_strToReplace, findStr);
  env->ReleaseStringUTFChars(_replString, replStr);

  return (newString);
}
```