```
 #include "pulkit_WinRegKeyName.h"
#include "pulkit_SystemInfo.h"
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#include <windows.h>

/*helper function to start enumeration of names */
static int startNameEnumeration(JNIEnv *env,jobject this_obj ,jclass this_class)
{

jfieldID id_index;
jfieldID id_count;
jfieldID id_root;
jfieldID id_path;
jfieldID id_hkey;
jfieldID id_maxsize;

HKEY root;
jstring path;
const char* cpath;
HKEY hkey;
DWORD maxsize= 0;
DWORD count =0;



/*get the field ID */
id_root =(*env)->GetFieldID(env,this_class,"root","I");
id_path =(*env)->GetFieldID(env,this_class,"path","Ljava/lang/String;");
id_hkey=(*env)->GetFieldID(env,this_class,"hkey","I");
id_maxsize =(*env)->GetFieldID(env,this_class,"maxsize","I");
id_index =(*env)->GetFieldID(env,this_class,"index","I");
id_count=(*env)->GetFieldID(env,this_class,"count","I");


/*get the field values */
root =(HKEY)(*env)->GetIntField(env,this_obj,id_root);
path=(jstring)(*env)->GetObjectField(env,this_obj,id_path);
cpath =(*env)->GetStringUTFChars(env,path,NULL);



/*open the regitsry key */
if (RegOpenKeyEx(root,cpath,0,KEY_READ,&hkey )!=ERROR_SUCCESS)
{
  (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),
                    "Open Key Failed");
   (*env)->ReleaseStringUTFChars(env,path,cpath);
   RegCloseKey(hkey);
    return -1;
}

(*env)->ReleaseStringUTFChars(env,path,cpath);


/*query num of name in key and maxlength of names */

if
(RegQueryInfoKey(hkey,NULL,NULL,NULL,NULL,NULL,NULL,&count,&maxsize,NULL,NULL,
NULL) !=ERROR_SUCCESS)

 {

  (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),"Query Failed");
  RegCloseKey(hkey);
  return -1;
  }

/*set the field values */
 (*env)->SetIntField(env,this_obj,id_hkey,(DWORD) hkey);
 (*env)->SetIntField(env,this_obj,id_maxsize,maxsize +1);
 (*env)->SetIntField(env,this_obj,id_index,0);
 (*env)->SetIntField(env,this_obj,id_count,count);
 return  count;

}


JNIEXPORT jboolean JNICALL Java_pulkit_WinRegKeyName_hasMoreElements
  (JNIEnv *env , jobject this_obj)

{


jclass this_class;
jfieldID id_index;
jfieldID id_count;
int index;
int count;

/*get the class */

this_class = (*env)->GetObjectClass(env,this_obj);
```

```
/*get the field IDs */
id_index = (*env)->GetFieldID(env,this_class,"index","I");
id_count =(*env)->GetFieldID(env,this_class,"count","I");

index =(*env)->GetIntField(env,this_obj,id_index);

 /*for first iteration */
if(index  == -1)
 {
  count =startNameEnumeration(env,this_obj,this_class);
  index =0;
  }
else
 count = (*env)->GetIntField(env,this_obj,id_count);
 return index < count;
}


JNIEXPORT jstring JNICALL Java_pulkit_WinRegKeyName_nextElement
  (JNIEnv *env, jobject this_obj)

  {

  jclass this_class ;
  jfieldID id_index;
  jfieldID id_hkey;
  jfieldID id_count;
  jfieldID id_maxsize;

  HKEY hkey;
  int index;
  int count;
  DWORD maxsize;

 char *cret;
  jstring ret ;

/*get the class */

this_class =(*env)->GetObjectClass(env,this_obj);

/*get the field IDs */

id_index =(*env)->GetFieldID(env,this_class,"index","I");
id_count =(*env)->GetFieldID(env,this_class,"count","I");
id_hkey =(*env)->GetFieldID(env,this_class,"hkey","I");
id_maxsize=(*env)->GetFieldID(env,this_class,"maxsize","I");
```

```
index =(*env)->GetIntField(env,this_obj,id_index);
/*for first time */

if(index== -1)
{
 count = startNameEnumeration(env,this_obj,this_class );
 index = 0;
  }

else
count = (*env)->GetIntField(env,this_obj,id_count);

if(index >=count) /*at end */
{
 (*env)->ThrowNew(env,(*env)->FindClass(env,"java/util/NoSuchElementException"),
 "past end of enumeration");
  return NULL;
}

maxsize = (*env)->GetIntField(env,this_obj,id_maxsize);
hkey =(HKEY)(*env)->GetIntField(env,this_obj,id_hkey);
cret = (char *)malloc(maxsize);

/*find next name */
if (RegEnumValue(hkey,index,cret,&maxsize,NULL,NULL,NULL,NULL)!=ERROR_SUCCESS)
{
  (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),
   "Enum value failed");
 free(cret);
RegCloseKey(hkey);
(*env)->SetIntField(env,this_obj,id_index,count);
return NULL;
}

ret =(*env)->NewStringUTF(env,cret);
free(cret);

/*increment index */
index++;
(*env)->SetIntField(env,this_obj,id_index,index);

if(index ==count)
{
 RegCloseKey(hkey);
}
return ret;
}
```

```
/*****************************************************************************
 Function Name :- getValue()
 Return            :-value of the subkey
*****************************************************************************/

JNIEXPORT jobject JNICALL Java_pulkit_SystemInfo_getValue
  (JNIEnv *env, jobject this_obj, jobject name)
 {


     const char* cname;
     jstring path;
     const char* cpath;
     HKEY hkey;

     DWORD type;
     DWORD size;
     jclass this_class;
     jfieldID id_root;
     jfieldID id_path;
     HKEY root;
     jobject ret;
     char* cret;

     /* get the class */
     this_class = (*env)->GetObjectClass(env,this_obj);

     /*get field id's */

     id_root =(*env)->GetFieldID(env,this_class,"root","I");
     id_path =(*env)->GetFieldID(env,this_class,"path","Ljava/lang/String;");


     /*get the fields */
     root =(HKEY) (*env)->GetIntField(env,this_obj,id_root);
     path =(jstring)(*env)->GetObjectField(env,this_obj,id_path);
     cpath =(*env)->GetStringUTFChars(env,path,NULL);


     /*open the regitsry key */
      if (RegOpenKeyEx(root,cpath,0,KEY_READ,&hkey )!=ERROR_SUCCESS)
     {
       (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),
                 "Open Key Failed");
        RegCloseKey(hkey);
        (*env)->ReleaseStringUTFChars(env,path,cpath);
         return NULL;
      }
```

```
(*env)->ReleaseStringUTFChars(env,path,cpath);

cname = (*env)->GetStringUTFChars(env,name,NULL);

/*find the type and size of the value */
if (RegQueryValueEx(hkey,cname,NULL,&type,NULL,&size) !=ERROR_SUCCESS)
 {
   (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),
            "Query Key Failed");
     RegCloseKey(hkey);
     (*env)->ReleaseStringUTFChars(env,name,cname);
   return NULL;


 }

   /*get memory to hold value */
  cret = (char*)malloc(size);

  /* read the value */
  if (RegQueryValueEx(hkey,cname,NULL,&type,cret,&size) !=ERROR_SUCCESS)
  {

     (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),
            "Query Key Failed");
      free(cret);
      RegCloseKey(hkey);
      (*env)->ReleaseStringUTFChars(env,name,cname);
     return NULL;

    }



  /*depending on the type ,store the value in a String ,integer,or a byte array */

  if (type == REG_SZ)
  {
     ret =(*env)->NewStringUTF(env,cret);
  }

  else if (type == REG_DWORD)
  {
     jclass class_Integer = (*env)->FindClass(env,"java/lang/Integer");

     /*get method Id */
     jmethodID id_Integer = (*env)->GetMethodID(env,class_Integer,"<init>","(I)V");
     int value =*(int*)cret;
```

```
            /*invoke onstructor */
            ret = (*env)->NewObject(env,class_Integer,id_Integer,value);
        }

        else if (type == REG_BINARY)
        {
         ret = (*env)->NewByteArray(env,size);
         (*env)->SetByteArrayRegion(env,(jarray) ret,0,size,cret);
        }

        else
        {

           (*env)->ThrowNew(env,(*env)->FindClass(env,"Win32RegKeyException"),"Unsupported
Value Type");
             RegCloseKey(hkey);
            (*env)->ReleaseStringUTFChars(env,name,cname);
            ret =NULL;
        }

        free(cret);
        RegCloseKey(hkey);
        (*env)->ReleaseStringUTFChars(env,name,cname);
        return ret;


   }

 /*****************************************************************************
   This function sets the value of subkey
   *****************************************************************************/
  JNIEXPORT void JNICALL Java_pulkit_SystemInfo_setValue
  (JNIEnv *env, jobject this_obj, jstring name, jobject value)
   {

      const char* cname;
      jstring path;
      const char* cpath;
      HKEY hkey;
      DWORD type;
      DWORD size;
       jclass this_class;
       jclass class_value;
       jclass class_Integer;
       jfieldID id_root;
       jfieldID id_path;
       HKEY root;
       const char* cvalue;
  int ivalue;

   /* get the class */
this_class = (*env)->GetObjectClass(env,this_obj);

/*get field id's */

id_root =(*env)->GetFieldID(env,this_class,"root","I");
id_path =(*env)->GetFieldID(env,this_class,"path","Ljava/lang/String;");


/*get the fields */
root =(HKEY)(*env)->GetIntField(env,this_obj,id_root);
path =(jstring)(*env)->GetObjectField(env,this_obj,id_path);
cpath =(*env)->GetStringUTFChars(env,path,NULL);


/*open the regitsry key */
 if (RegOpenKeyEx(root,cpath,0,KEY_WRITE,&hkey )!=ERROR_SUCCESS)
{
   (*env)->ThrowNew(env,(*env)->FindClass(env,"WinRegKeyException"),
              "Open Key Failed");
    RegCloseKey(hkey);
    (*env)->ReleaseStringUTFChars(env,path,cpath);
     return ;
  }

  (*env)->ReleaseStringUTFChars(env,path,cpath);

  cname = (*env)->GetStringUTFChars(env,name,NULL);



  class_value =(*env)->GetObjectClass(env,value);
  class_Integer=(*env)->FindClass(env,"java/lang/Integer");

  /*etermine the type of the value being passed*/

  if ((*env)->IsAssignableFrom(env,class_value,(*env)->FindClass(env,"java/lang/String")))
  {

      /* it is a string*/
      cvalue =(*env)->GetStringUTFChars(env,(jstring) value,NULL);
      type=REG_SZ;
      size =(*env)->GetStringLength(env,(jstring)value) + 1;

  }
  else if ((*env)->IsAssignableFrom(env,class_value,class_Integer))
  {
        /* it is an integer*/
        jmethodID id_intValue =(*env)->GetMethodID(env,class_Integer,"intValue","()I");
        ivalue = (*env)->CallIntMethod(env,value,id_intValue);
        type =REG_DWORD;
        cvalue =(char*)&ivalue;
        size = 4;

      }

      else if ((*env)->IsAssignableFrom(env,class_value,(*env)->FindClass(env,"[B")))
      {
        /* it is a byte array */
        type =REG_BINARY;
        cvalue = (char*)(*env)->GetByteArrayElements(env,(jarray)value,NULL);
        size =(*env)->GetArrayLength(env,(jarray)value);
      }

       else
       {
        /* default condition to handle unknown type  */
           (*env)->ThrowNew(env,(*env)->FindClass(env,"Win32RegKeyException"),"Unsupported
Value Type");
           RegCloseKey(hkey);
           (*env)->ReleaseStringUTFChars(env,name,cname);
          return;

       }

      /* set the value  */
       if (RegSetValueEx(hkey,cname,0,type,cvalue,size) != ERROR_SUCCESS)
       {
         (*env)->ThrowNew(env,(*env)->FindClass(env,"Win32RegKeyException"),"Value can not be
updated");
           RegCloseKey(hkey);
           (*env)->ReleaseStringUTFChars(env,name,cname);
       }
         RegCloseKey(hkey);
         (*env)->ReleaseStringUTFChars(env,name,cname);

       /*relaese pointer for string and byte array */
       if (type == REG_SZ)
       {

          (*env)->ReleaseStringUTFChars(env,(jstring)value,cvalue);
       }
        else if (type ==REG_BINARY)
        {
         (*env)->ReleaseByteArrayElements(env,(jarray) value,(jbyte*) cvalue,0);
        }
```

```
}
```