1.-

Descargar Java 1.8

Descargar Android SDK

Descargar Appium

Intalar Java 1.8

Configurar variable de entorno de Java home

Instalar Android SDK

Configurar variable de entorno Android home

Configurar emulador Android

Instalar appium

Ejecutar emulador Android

Ejecutar appium

Configurar capabilities

Iniciar ejecución.

Fin.

2.- **Clase**: contenedor del mundo real llevado a la POO para simular un objeto y ser utilizado para la ejecución de construcciones de software.

**Método**: Sección de código que tiene como finalidad ejecutar cierto proceso programado dentro de una clase. Estos pueden manejar argumentos de entrada y salidas o sólo ejecuciones de un solo fin sin salidas.

**Objeto**: Se crea a partir de la instancia de una clase, se realiza la carga en memoria de dicho objeto con sus propiedades tales como campos, métodos y variables.

**Herencia**: Consta de heredar o ceder propiedades ya definidas de una clase, como campos o métodos a otra para la reutilización de código y la agilización de desarrollo.

**Instanciar**: Teniendo una clase definida se realiza la instanciación que no es más cargar un objeto en memoria de dicha clase para su utilización en un fin definido.

**Atributos**: Características que se pueden definir dentro de una clase, estos pueden ser públicos, privados o protegidos.

3.- Public Class CampoFutbol extends Estadio{ 🡨------Definicion de clase y herencia.

Int largoCampo; 🡨-----atributo

Int anchoCampo; 🡨-----atributo

Public int calculaAreaCancha(int x, int y){ 🡨-----Metodo

largoCampo = x;

anchoCampo = y;

Return largoCampo \* anchoCampo;

}

Public static void main(String [] args){

CampoFutbol cf = new CampoFutbol(); 🡨----Instanciar

cf.calculaAreaCancha(4,5)

}

El objeto no se puede representar porque se encuentra cargado en memoria de la máquina.

4.- Herramienta que sirve como repositorio de bibliotecas que pueden ser utilizadas para proyectos Web, aparte de que ofrece reglas o normas para dicha construcción.

5.- mvn archetype:generate -DgroupId=com.cursoAppium.app -DartifactId=projectMavenAlumnos -DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4 -DinteractiveMode=false

6.-El archivo pom es el archivo principal para configurar nuestro proyecto Maven.

groupID: paquete principal del proyecto.

artifactID: nombre del proyecto.

Versión: versión inicial o actual del proyecto.

Dependencias: etiqueta que contendrá todas las dependencias necesarias para el proyecto.

Dependency: biblioteca especifica que se puede bajar al proyecto colocando el formato adecuado para bajarlas.

Profiles: etiqueta que marca el inicio de los profiles definidos para el proyecto Maven.

Profile: configuración especificada para dicho perfil, si se elige algún perfil en específico cargara las configuraciones y ejecuciones para el proyecto Maven.

Properties. En esta sección se pueden definir varibles para ser utilizandos en el pom, como por ejemplo la versión java o Maven.

7.- maven clean.

8.- maven compile.

9.- maven install

10.- mvn install -P appiumget\_apptive

11.- Es una herramienta que sirve como repositorio de código o proyectos en general, tiene la utilidad de poder compartir código de manera simple, aparte en proyectos grandes muchos usuarios pueden aportar avances al mismo proyecto.

12.- herramienta que sirve como almacén de archivos principalmente de proyectos de software.

13.- Git clone.

14.- git commit

15.-git branch

16.- git checkout -b

17.-git push

18.- Subir avances del proceso de manera recurrente, sino uno se atrasa subiendo cambios y luego es un desastre y se pisan cambios.

19.- getById, getByName, getByClass, , getByPath

20.- Marco de trabajo que define reglas o normas para la creación o definición de algo, dígase un proyecto por ejemplo.

21.- Framework definido para pruebas móviles que define los pasos para la creación de clases.

22.-

Package: paquete donde se encuentra definidia y creada la clase.

Import: se importan las bibleotecas necesarias que utiliza la clase.

Public class.- se define el nombre de la clase y se hereda de la clase DriverConfiguration

Final static logger: se define un campo de logger para permitir la impresión en la consola.

Se define un campo de tipo AndroidDriver.

Se define un método prepareAndroidAppium que se recibe un argumento de tipo String y define de igual forma una excepción en dado caso de error.

Se define un campo de tipo File que sirve para crear en memoria el espacio necesario que se reserva para recibir un archivo y se le deben enviar las propiedades de ubicación de archivo y nombre de archivo.

Se define una variable de tipo DesiredCapabilities para prepararel seteo de campos.

Se setea una capability donde se envía el nomnre de la plataforma en este caso Android.

Se setea una capability donde se coloca la versión de Android hemos ocupado 9.

Se setea una capability donde se coloca el nombre o Id del dispositivo.

Se setea una capability donde se coloca el paquete principal de la aplicación.

Se setea una capability donde se coloca cuál es la clase principal con todo y paquete.

Se setea una capability donde se colocan si se otorgan permisos o no.

Se setea una capability donde se coloca la ubicación de la apk.

Se instancia el controlador con todo lo anterior enviando el objeto URLAppium y todas las capabilities definidas previamente.

Se define un método que regresa un androidDirver. Este método sirve para que ya construido el driver se pueda regresar a quien lo haya llamado.