





























Main.java

package org.example;

public class Main {

public static void main(String[] args) {

// To avoid making changes for both Cucumber Java and actual Test.java (the file which uses Cucumber's Feature - Scenario - Given/Then/When file)

}

}

build.gradleplugins **{**

**plugins {**

**id 'java'**

**}**

**group = 'org.example'**

**version = '1.0-SNAPSHOT'**

**repositories {**

**mavenCentral()**

**maven { url = uri("https://jitpack.io")}**

**google()**

**gradlePluginPortal()**

**mavenLocal()**

**flatDir {**

**dirs "libs"**

**}**

**}**

**dependencies {**

**implementation 'io.cucumber:cucumber-junit:7.22.1'**

**implementation 'org.junit.jupiter:junit-jupiter-api:5.12.2'**

**implementation 'org.junit.jupiter:junit-jupiter-engine:5.12.2'**

**implementation 'io.cucumber:cucumber-core:7.22.1'**

**implementation 'io.cucumber:cucumber-spring:7.22.1'**

**implementation 'org.apache.groovy:groovy-all:4.0.26'**

**implementation 'io.cucumber:cucumber-gherkin:7.22.1'**

**implementation 'io.cucumber:gherkin:32.1.1'**

**implementation 'io.cucumber:gherkin-utils:9.2.0'**

**implementation 'io.cucumber:cucumber-jvm-groovy:6.10.4'**

**implementation 'io.cucumber:cucumber-groovy:6.10.4'**

**implementation 'io.cucumber:cucumber-java:7.22.1'**

**implementation 'org.seleniumhq.selenium:selenium-chrome-driver:4.32.0'**

**implementation 'com.aventstack:extentreports:5.1.2';**

**implementation 'org.springframework:spring-beans:6.2.6'**

**implementation 'org.springframework:spring-core:6.2.6'**

**implementation 'org.springframework:spring-web:6.2.6'**

**implementation 'org.springframework:spring-test:6.2.6'**

**implementation 'org.springframework:spring-webmvc:6.2.6'**

**implementation 'org.springframework:spring-jms:6.2.6'**

**implementation 'org.springframework:spring-messaging:6.2.6'**

**implementation 'org.springframework:spring-aop:6.2.6'**

**implementation 'org.springframework:spring-orm:6.2.6'**

**implementation 'org.springframework:spring-context-support:6.2.6'**

**implementation 'org.springframework:spring-context:6.2.6'**

**implementation 'org.springframework:spring-tx:6.2.6'**

**implementation 'org.springframework:spring-jdbc:6.2.6'**

**}**

**test {**

**useJUnitPlatform()**

TesDesktopWebsites.java

package hellocucumber;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.MediaEntityBuilder;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.reporter.ExtentSparkReporter;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import io.cucumber.spring.CucumberContextConfiguration;

import org.junit.runner.RunWith;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.io.FileHandler;

import java.io.File;

import java.io.IOException;

import java.util.List;

import java.time.LocalDateTime;

import java.util.Objects;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or

// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.

@CucumberContextConfiguration

@RunWith(Cucumber.class)

@CucumberOptions(

// features should start with "src/test/resources/features/xxxxxxxxxxx.feature",

features = "src/test/resources/features/Selenium\_Tutorial.feature",

// glue starts with "src/test/java/#package for the Test Java class with the Given;When;Then annotations#/",

glue={"src/test/java/hellocucumber/"},

// plugin = = {"pretty", "html:target/cucumber-reports"}

plugin = {"pretty", "html:target/cucumber-reports"}

)

public class TestDesktopWebsites {

// Test.java class should be within src > test

WebDriver driver1 = new ChromeDriver();

String websiteTitle = "";

String capturedScreenshotImageFilepathString = "";

String currentLocalDateTimeForExtentSparkReporter = *currentLocalDateTimeWithDdMmYyFormat*();

ExtentReports extentReport = new ExtentReports();

// should be target/Spark.html, not an actual filepath starting from C:\\

ExtentSparkReporter extentSparkReporter = new ExtentSparkReporter("target/Spark\_" + currentLocalDateTimeForExtentSparkReporter + ".html");

ExtentTest extentTest = extentReport.createTest("Selenium\_Tutorial Test");

FirstGroupOfWebsites firstGroupOfWebsites;

SecondGroupOfWebsites secondGroupOfWebsites;

ThirdGroupOfWebsites thirdGroupOfWebsites;

public TestDesktopWebsites() throws IOException {

}

// Note that this method is not executed, if running the feature with the Given, When, Then lines

void testMethod() {

System.*out*.println("testing the IntelliJ IDEA - Help - Testing - Create Tests - Right-click to generate Test Method for JUnit5.");

}

@org.junit.Test

@org.junit.jupiter.api.Test

@Given("that the easiest way to find WebElements is by driver.findElement; and driver.findElements which returns a List WebElement")

public void testMethod1() throws IOException {

// this line only works without an error, in a public method, as it is a public void method

// the other three lines (originally above this fourth line) are public methods

extentReport.attachReporter(extentSparkReporter);

FirstGroupOfWebsites firstGroupOfWebsites = new FirstGroupOfWebsites((ChromeDriver) driver1, extentTest, extentSparkReporter, extentReport);

System.*out*.println("testing");

}

@org.junit.Test

@org.junit.jupiter.api.Test

@When("it is not possible to do so for certain WebElements in this way")

public void testMethod2 () throws IOException {

SecondGroupOfWebsites secondGroupOfWebsites = new SecondGroupOfWebsites((ChromeDriver) driver1, extentTest, extentSparkReporter, extentReport);

}

@org.junit.Test

@org.junit.jupiter.api.Test

@Then("we use driver.findElement By.tagName\"...\", for which ... can be i for <i> or button for <button>")

public void testMethod3 () throws IOException {

ThirdGroupOfWebsites thirdGroupOfWebsites = new ThirdGroupOfWebsites((ChromeDriver) driver1, extentTest, extentSparkReporter, extentReport);

}

// This method is for the filename for capturing screenshot images which are not in the ExtentReport

public static String takeScreenshot(WebDriver driver) throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

TakesScreenshot screenshot = ((TakesScreenshot) driver);

File sourceFile = screenshot.getScreenshotAs(OutputType.*FILE*);

File destinationFile = new File("C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png");

FileHandler.*copy*(sourceFile, destinationFile);

return "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png";

}

// This method is for returning the substring of the current LocalDateTome, for the filename of the ExtentReport

public static String currentLocalDateTimeWithDdMmYyFormat() throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

// There seems to be an error for which the two digits for day are incorrect, so localDateTime.getDayOfmonth is used(), instead of dayFirstDigitChar and daySecondDigitChar

return localDateTime.getDayOfMonth() + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar;

}

}

FirstGroupOfWebsites.java

package hellocucumber;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.MediaEntityBuilder;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.reporter.ExtentSparkReporter;

import org.apache.tools.ant.types.resources.First;

import org.openqa.selenium.\*;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.io.FileHandler;

import java.io.File;

import java.io.IOException;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Objects;

import static hellocucumber.TestDesktopWebsites.*takeScreenshot*;

public class FirstGroupOfWebsites {

String websiteTitle = "";

String capturedScreenshotImageFilepathString = "";

public FirstGroupOfWebsites (ChromeDriver driver, ExtentTest extentTest, ExtentSparkReporter extentSparkReporter, ExtentReports extentReport) throws IOException {

System.*out*.println("testing");

String capturedScreenshotImageFilepathString = new String("");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

driver.get("https://demo.guru99.com/");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

// The web browser tab's title (depends on the website)

String websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Guru99 Bank Home Page")) {

// requires actual filepath of takeScreenshot(WebDriver object)

System.*out*.println("The current web browser tab title is " + "\"" + websiteTitle + "\"; which should be \"Guru99 Bank Home Page\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is " + "\"" + websiteTitle + "\"; which should be \"Guru99 Bank Home Page\".");

} else {

System.*out*.println("The current web browser tab title cannot be retrieved; which should be \"Guru99 Bank Home Page\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title cannot be retrieved; which should be \"Guru99 Bank Home Page\".");

}

WebElement emailInputField = driver.findElement(By.*name*("emailid"));

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

// check for empty String for email input field value

//String emailInputFieldText = emailInputField.getText();

// Only for testing purposes, to ensure that WebElement object.getText() works

// Capture screenshot image, and input field text value immediately after sending the String to the input field

if (emailInputField.isDisplayed()) {

emailInputField.sendKeys("abc@gmail.com");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

}

// actual verification for pass or fail

// check for email input field value being "abc@gmail.com"

// emailInputFieldText = emailInputField.getText();

// Note: Text input field cannot allow Selenium to find the text with WebElement object.getText();

if (emailInputField.isDisplayed()) {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The email input field value's text is the correct \"abc@gmail.com\".");

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The email input field value's text is not \"abc@gmail.com\".");

}

String emailText = "abc@gmail.com";

int emailLength = emailText.length();

// To use backspace to clear the initially entered emailText

for (int i = 0; i < emailLength; i++) {

emailInputField.sendKeys(Keys.*BACK\_SPACE*);

}

// check for the email input field value being ""

// emailInputFieldText = emailInputField.getText();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

if (emailInputField.isDisplayed()) {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The email input field value's text is the correct \"\".");

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The email input field value's text is not \"\".");

}

//emailInputFieldText = emailInputField.getText();

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The email input field value \"abc@gmail.com\" has been removed by the Backspace key.");

emailInputField.sendKeys("abc@gmail.com");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

//emailInputFieldText = emailInputField.getText();

// check for email input field value being "abc@gmail.com"

if (emailInputField.isDisplayed()) {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The email input field value's text is the correct \"abc@gmail.com\".");

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The email input field value's text is not \"abc@gmail.com\".");

}

// The Submit button

WebElement submitButton = driver.findElement(By.*name*("btnLogin"));

if (submitButton.isDisplayed()) {

submitButton.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

}

// The web browser tab's title (depends on the website)

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Guru99 Bank Home Page")) {

// requires actual filepath of takeScreenshot(WebDriver object)

System.*out*.println("The current web browser tab title is " + "\"" + websiteTitle + "\"; which should be \"Guru99 Bank Home Page\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is " + "\"" + websiteTitle + "\"; which should be \"Guru99 Bank Home Page\".");

} else {

System.*out*.println("The current web browser tab title cannot be retrieved; which should be \"Guru99 Bank Home Page\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title cannot be retrieved; which should be \"Guru99 Bank Home Page\".");

}

driver.get("https://www.bing.com");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

// The web browser tab's title (depends on the website)

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Search - Microsoft Bing")) {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; which should be \"Search - Microsoft Bing\".");

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title is the incorrect" + "\"" + websiteTitle + "\"; which should be \"Search - Microsoft Bing\".");

}

// Microsoft Bing search bar

WebElement microsoftBingSearchBar = driver.findElement(By.*name*("q"));

// String microsoftBingSearchBarText = microsoftBingSearchBar.getText();

// Selenium does not allow text input field value to be retrieved with WebElement object.getText();

// Search bar should initially have an empty String

if (microsoftBingSearchBar.isDisplayed()) {

microsoftBingSearchBar.sendKeys("MapleStorySEA Unfunded Amino");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

}

if (microsoftBingSearchBar.isDisplayed()) {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search term input is the correct \"MapleStorySEA Unfunded Amino\".");

microsoftBingSearchBar.sendKeys(Keys.*ENTER*);

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Microsoft Bing search term input is the incorrect \"MapleStorySEA Unfunded Amino\".");

microsoftBingSearchBar.sendKeys(Keys.*ENTER*);

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

}

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

// Microsoft Bing - First Page of search results - Default All tab

if (Objects.*equals*(websiteTitle, "MapleStorySEA Unfunded Amino - Search")) {

System.*out*.println("The current web browser tab title is " + "\"" + websiteTitle + "\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search has been triggered, to reach the webpage with the web browser tab title \"MapleStorySEA Unfunded Amino - Search\".");

} else {

System.*out*.println("The current web browser tab title cannot be retrieved.");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Microsoft Bing search has not been triggered, to reach the webpage with the web browser tab title \"MapleStorySEA Unfunded Amino - Search\".");

}

// MapleStory Unfunded Amino (Microsoft Bing search results - Search tab)

//websiteTitle = driver.getTitle();

// System.out.println("The website title is \"" + websiteTitle + "\".");

// Search engine result Search tab

// id is based on the line above highlighted text for inspected element

// Still on the same initial Search Results webpage

// Should only check for whether the WebElement searchTab is displayed

WebElement searchTab = driver.findElement(By.*id*("b-scopeListItem-copilotsearch"));

if (searchTab.isDisplayed()) {

searchTab.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search results - Search tab has been reached.");

} else {

searchTab.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Microsoft Bing search results - Search tab has not been reached.");

}

// Go back to All tab from Search tab webpage

// id is based on the line above highlighted text for inspected element

WebElement allTab = driver.findElement(By.*id*("b-scopeListItem-web"));

if (allTab.isDisplayed()) {

allTab.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search results - All tab has been reached.");

} else {

allTab.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Microsoft Bing search results - All tab has not been reached.");

}

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "MapleStorySEA Unfunded Amino - Search")) {

System.*out*.println("The current web browser tab title is " + "\"" + websiteTitle + "\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search results webpage has been reached.");

} else {

System.*out*.println("The current web browser tab title cannot be retrieved.");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Microsoft Bing search results webpage has not been reached.");

}

// web browser search engine result uses partialLinkText is not working, due to YouTube videos being displayed at the right sometimes

//WebElement requiredSearchResultLink = driver.findElement(By.partialLinkText("Featured | [MapleStorySEA] Unfunded Tips Amino - Amino Apps"));

// Due to video section appearing at the right

// Which causes less text for the Microsoft Bing search results hyperlink to be displayed

WebElement requiredSearchResultLink = driver.findElement(By.*partialLinkText*("Featured | [MapleStorySEA] Unfunded Tips"));

// Use tagName "a" instead

// Cannot use tagName, keeps changing position in List <WebElement>

//List <WebElement> tagNameAWebElementList = driver.findElements(By.tagName("a"));

//WebElement requiredSearchResultLink = tagNameAWebElementList.get(30);

// String requiredSearchResultPartialLinkTextString = requiredSearchResultPartialLinkText.getText();

//String requiredSearchResultLinkText = requiredSearchResultLink.getText();

//if (requiredSearchResultPartialLinkText.isDisplayed() && requiredSearchResultPartialLinkTextString.equals("Featured | [MapleStorySEA] Unfunded Tips Amino - Amino Apps")) {

if (requiredSearchResultLink.isDisplayed()) {

for (int i = 0; i < 50; i++) {

//requiredSearchResultPartialLinkText.sendKeys(Keys.DOWN);

requiredSearchResultLink.sendKeys(Keys.*DOWN*);

}

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search results - bottom of the first page been reached; by using the WebElement for requiredSearchResultLink \"Featured | [MapleStorySEA] Unfunded Tips Amino - Amino Apps\".");

for (int i = 0; i < 50; i++) {

//requiredSearchResultPartialLinkText.sendKeys(Keys.UP);

requiredSearchResultLink.sendKeys(Keys.*UP*);

}

*takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Microsoft Bing search results - top of the first page has been reached; by using the WebElement for requiredSearchResultLink \"Featured | [MapleStorySEA] Unfunded Tips Amino - Amino Apps\".");

//requiredSearchResultPartialLinkText.click();

requiredSearchResultLink.click();

//}

}

// Problem: Search engine search result opens in a new web browser tab

// New ChromeDriver object needs to be created to open a new web browser window

driver.get("https://aminoapps.com/c/maplestorysea/home/");

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino website has been reached.");

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Featured | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

}

WebElement aminoSearchBar = driver.findElement(By.*className*("nav-search-input"));

//String aminoSearchBarText = aminoSearchBar.getText();

if (aminoSearchBar.isDisplayed()) {

aminoSearchBar.sendKeys("MapleStory Unfunded Amino");

// non-empty input has been entered

// Text input field value cannot be found by Selenium

//aminoSearchBarText = aminoSearchBar.getText();

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino search term has been provided as the correct \"MapleStorySEA Unfunded Amino\".");

} else {

aminoSearchBar.sendKeys("MapleStory Unfunded Amino");

// non-empty input has been entered

// aminoSearchBarText = aminoSearchBar.getText();

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino search term has been provided as not \"MapleStorySEA Unfunded Amino\".");

}

String aminoSearchTerm = "MapleStorySEA Unfunded Amino";

int aminoSearchTermLength = aminoSearchTerm.length();

for (int i = 0; i < aminoSearchTermLength; i++) {

aminoSearchBar.sendKeys(Keys.*BACK\_SPACE*);

}

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

//aminoSearchBarText = aminoSearchBar.getText();

//if (aminoSearchBarText.equals("")) {

//extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.createScreenCaptureFromPath(capturedScreenshotImageFilepathString).build()).log(Status.PASS, "The Amino search term has been removed by the Backspace key; and the current Amino search term is \"" + aminoSearchBarText + "\".");

//} else {

//extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.createScreenCaptureFromPath(capturedScreenshotImageFilepathString).build()).log(Status.FAIL, "The Amino search term has not been removed by the Backspace key; and the current Amino search term is \"" + aminoSearchBarText + "\".");

//}

aminoSearchBar.sendKeys("MapleStorySEA Unfunded Amino");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

//aminoSearchBarText = aminoSearchBar.getText();

//if(aminoSearchBarText.equals("MapleStorySEA Unfunded Amino")) {

//extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.createScreenCaptureFromPath(capturedScreenshotImageFilepathString).build()).log(Status.PASS, "The Amino search term has been re-provided as the correct \"" + aminoSearchBarText + "\", which should be \"MapleStorySEA Unfunded Amino\".");

//} else {

//extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.createScreenCaptureFromPath(capturedScreenshotImageFilepathString).build()).log(Status.FAIL, "The Amino search term has been re-provided as the incorrect \"" + aminoSearchBarText + "\", which should be \"MapleStorySEA Unfunded Amino\".");

//}

aminoSearchBar.sendKeys(Keys.*ENTER*);

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

//websiteTitle = driver.getTitle();

List<WebElement> tagNameH3WebElementList = driver.findElements(By.*tagName*("h3"));

WebElement popularPostsText = tagNameH3WebElementList.get(0);

String popularPostsTextString = popularPostsText.getText();

// This test step usually fails due to changing web browser tab title

// So changed to use fixed non-hyperlink text "Popular Posts", for the "Popular Posts" section

// For this case, the tag is h3

System.*out*.println("The Popular Posts' section's Popular Posts title for entering a search term from an Amino community is \"" + popularPostsTextString + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(popularPostsTextString, "Popular posts")) {

System.*out*.println("The Popular Posts' section's Popular Posts title for entering a search term from an Amino community is the correct " + "\"" + popularPostsTextString + "\"; which should be \"Popular Posts\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Popular Posts' section's Popular Posts title for entering a search term from an Amino community is the correct " + "\"" + popularPostsTextString + "\"; which should be \"Popular Posts\".");

} else {

System.*out*.println("The Popular Posts' section's Popular Posts title for entering a search term from an Amino community is the incorrect " + "\"" + popularPostsTextString + "\"; which should be \"Popular Posts\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Popular Posts' section's Popular Posts title for entering a search term from an Amino community is the correct " + "\"" + popularPostsTextString + "\"; which should be \"Popular Posts\".");

}

driver.get("https://aminoapps.com/c/maplestorysea/home/");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Featured | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is " + "\"" + websiteTitle + "\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage is the correct \"" + websiteTitle + "\"; which should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title cannot be retrieved.");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage is the incorrect \"" + websiteTitle + "\"which should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".\");.");

}

WebElement privacyNoticeAcceptButton = driver.findElement(By.*className*("confirm-close"));

if (privacyNoticeAcceptButton.isDisplayed()) {

privacyNoticeAcceptButton.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The privacy notice pop-up's Accept button has been clicked on, so the pop-up has been closed.");

} else {

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The privacy notice pop-up's Accept button has not been clicked on.");

}

// findElement is only for the first matching object for the case of multiple objects with the same name/id/className

WebElement freeCharacterSlotExpansionCoupon = driver.findElement(By.*className*("overflow-hidden"));

if (freeCharacterSlotExpansionCoupon.isDisplayed()) {

freeCharacterSlotExpansionCoupon.click();

}

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Free Character Slot Expansion Coupon? | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is " + "\"" + websiteTitle + "\".");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage is the correct \"" + websiteTitle + "\"; which should be \"Free Character Slot Expansion Coupon? | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title cannot be retrieved.");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage is the incorrect \"" + websiteTitle + "\"; which should be \"Free Character Slot Expansion Coupon? | [MapleStorySEA] Unfunded Tips Amino\".");

}

driver.get("https://aminoapps.com/c/maplestorysea/home/");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage has been reached.");

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Featured | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; which should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; which should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; which should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; which should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

}

// findElements is for more than one matching object with the same class/name/id/partialLinkText which cannot be uniquely identified

// Base this on the Inspect - Ctrl+F order or position, search with "" marks

// className for this case is "overflow-hidden"

List<WebElement> fourOverflowHiddenClassnameWebElementList = driver.findElements(By.*className*("overflow-hidden"));

// Spell Trace and Star Force Enhancement

WebElement requiredThirdOverflowHiddenClassnameElement = fourOverflowHiddenClassnameWebElementList.get(2);

if (requiredThirdOverflowHiddenClassnameElement.isDisplayed()) {

requiredThirdOverflowHiddenClassnameElement.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage has been reached.");

} else {

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage has not been reached.");

}

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Spell Trace and Star Force Enhancement? | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Spell Trace and Star Force Enhancement? | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Spell Trace and Star Force Enhancement? | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Spell Trace and Star Force Enhancement? | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Spell Trace and Star Force Enhancement? | [MapleStorySEA] Unfunded Tips Amino\".");

}

driver.get("https://aminoapps.com/c/maplestorysea/home/");

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Featured | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

}

// findElements is for more than one matching object with the same class/name/id/partialLinkText which cannot be uniquely identified

// Base this on the Inspect - Ctrl+F order or position, search with "" marks

// className for this case is "overflow-hidden"

List<WebElement> sevenLabelClassnameWebElementList = driver.findElements(By.*className*("label"));

// Spell Trace and Star Force Enhancement

WebElement requiredSecondLabelClassnameElement = sevenLabelClassnameWebElementList.get(1);

if (requiredSecondLabelClassnameElement.isDisplayed()) {

requiredSecondLabelClassnameElement.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage has been reached.");

} else {

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage has not been reached.");

}

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Latest | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Latest | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The current web browser tab title is the correct " + "\"" + websiteTitle + "\"; and should be \"Latest | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Latest | [MapleStorySEA] Unfunded Tips Amino\".");

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The current web browser tab title is the incorrect " + "\"" + websiteTitle + "\"; and should be \"Latest | [MapleStorySEA] Unfunded Tips Amino\".");

}

}

// This method is for the filename for capturing screenshot images which are not in the ExtentReport

public static String takeScreenshot(WebDriver driver) throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

TakesScreenshot screenshot = ((TakesScreenshot) driver);

File sourceFile = screenshot.getScreenshotAs(OutputType.*FILE*);

File destinationFile = new File("C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png");

FileHandler.*copy*(sourceFile, destinationFile);

return "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png";

}

// This method is for returning the substring of the current LocalDateTome, for the filename of the ExtentReport

public static String currentLocalDateTimeWithDdMmYyFormat() throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

// There seems to be an error for which the two digits for day are incorrect, so localDateTime.getDayOfmonth is used(), instead of dayFirstDigitChar and daySecondDigitChar

return localDateTime.getDayOfMonth() + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar;

}

}

SecondGroupOfWebsites.java

package hellocucumber;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.MediaEntityBuilder;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.reporter.ExtentSparkReporter;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.io.FileHandler;

import java.io.File;

import java.io.IOException;

import java.time.LocalDateTime;

import static hellocucumber.TestDesktopWebsites.*takeScreenshot*;

public class SecondGroupOfWebsites {

String websiteTitle = "";

String capturedScreenshotImageFilepathString = "";

public SecondGroupOfWebsites (ChromeDriver driver, ExtentTest extentTest, ExtentSparkReporter extentSparkReporter, ExtentReports extentReport) throws IOException {

// For displaying a sentence for ExportReport

driver.get("https://aminoapps.com/c/maplestorysea/recent/");

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "When it is not possible to retrieve WebElements using name/id/className/partialLinkText.");

System.*out*.println("It is not possible to do so for certain WebElements in this way.");

}

// This method is for the filename for capturing screenshot images which are not in the ExtentReport

public static String takeScreenshot(WebDriver driver) throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

TakesScreenshot screenshot = ((TakesScreenshot) driver);

File sourceFile = screenshot.getScreenshotAs(OutputType.*FILE*);

File destinationFile = new File("C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png");

FileHandler.*copy*(sourceFile, destinationFile);

return "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png";

}

// This method is for returning the substring of the current LocalDateTome, for the filename of the ExtentReport

public static String currentLocalDateTimeWithDdMmYyFormat() throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

// There seems to be an error for which the two digits for day are incorrect, so localDateTime.getDayOfmonth is used(), instead of dayFirstDigitChar and daySecondDigitChar

return localDateTime.getDayOfMonth() + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar;

}

}

ThirdGroupOfWebsites.java

package hellocucumber;

import com.aventstack.extentreports.ExtentReports;

import com.aventstack.extentreports.ExtentTest;

import com.aventstack.extentreports.MediaEntityBuilder;

import com.aventstack.extentreports.Status;

import com.aventstack.extentreports.reporter.ExtentSparkReporter;

import org.openqa.selenium.\*;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.io.FileHandler;

import java.io.File;

import java.io.IOException;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Objects;

import static hellocucumber.TestDesktopWebsites.*takeScreenshot*;

public class ThirdGroupOfWebsites {

String websiteTitle = "";

String capturedScreenshotImageFilepathString = "";

public ThirdGroupOfWebsites (ChromeDriver driver, ExtentTest extentTest, ExtentSparkReporter extentSparkReporter, ExtentReports extentReport) throws IOException {

// Therefore, the Cucumber Feature version requires driver4 to be initialised for the Then method

driver.get("https://aminoapps.com/c/maplestorysea/recent/");

// i is a valid tagName, which is displayed starting with "<i "

List<WebElement> tagNameIWebElementList = driver.findElements(By.*tagName*("i"));

WebElement aminoHomeIcon = tagNameIWebElementList.get(2);

if (aminoHomeIcon.isDisplayed()) {

aminoHomeIcon.click();

}

websiteTitle = driver.getTitle();

System.*out*.println("The website title is \"" + websiteTitle + "\".");

// Note: Not all web browser tab titles work

if (Objects.*equals*(websiteTitle, "Featured | [MapleStorySEA] Unfunded Tips Amino")) {

System.*out*.println("The current web browser tab title is the correct \"" + websiteTitle + "\"; and should be the correct \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

} else {

System.*out*.println("The current web browser tab title is the incorrect \"" + websiteTitle + "\"; and should be the correct \"Featured | [MapleStorySEA] Unfunded Tips Amino\".");

}

// button is a valid tagName, which is displayed starting with "<button "

// testing for the Create Post button - pop-up

List<WebElement> tagNameButtonWebElementList = driver.findElements(By.*tagName*("button"));

WebElement createPostButton = tagNameButtonWebElementList.get(1);

if(createPostButton.isDisplayed()) {

createPostButton.click();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

//actual pop-up cannot be captured, can only prove that the pop-up was triggered by clicking on the Create Post button

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage's Create Post pop-up is displayed.");

System.*out*.println("Create Post button was clicked on to trigger the pop-up. The pop-up is not displayed at this point in time, but is displayed later.");

} else {

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage's Create Post button was not clicked on, so the Create Post pop-up is not displayed later.");

}

// trying to capture pop-up UI in ExtentReport

// So not possible to provide not working if else - else

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage's Create Post pop-up is displayed now.");

// Test for whether non-text-input field text can be retrieved

WebElement tagLine = driver.findElement(By.*className*("tagline"));

String taglineText = tagLine.getText();

// Captured screenshot image filename to be used for ExtentReport test step - Pass; or Fail

capturedScreenshotImageFilepathString = *takeScreenshot*(driver);

if(taglineText.equals("MapleStorySEA follows the original Korea MapleStory (by Nexon Korea) 's content.")){

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage's tagline text is displayed as the correct \"" + taglineText + "\"; which should be \"MapleStorySEA follows the original Korea MapleStory (by Nexon Korea) 's content.\".");

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage's tagline text is displayed as the incorrect \"" + taglineText + "\"; which should be \"MapleStorySEA follows the original Korea MapleStory (by Nexon Korea) 's content.\".");

}

// Testing if "Explore your interests and " for the pop-up can be accessed

// tagName is "p"

List <WebElement> tagNamePWebElementList = driver.findElements(By.*tagName*("p"));

WebElement exploreYourInterestsAndFindYourCommunitiesFullstopText = tagNamePWebElementList.get(1);

String exploreYourInterestsAndFindYourCommunitiesFullstopTextString = exploreYourInterestsAndFindYourCommunitiesFullstopText.getText();

// Note: Even though .getText() works, which seems to provide the same text

// This has been verified by the system to not match

// So this has to be tested by checking for whether the WebElement object is displayed

//Explore your interests and find your communities.

if(exploreYourInterestsAndFindYourCommunitiesFullstopText.isDisplayed()){

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*PASS*, "The Amino webpage's tagline text is displayed as the correct \"" + exploreYourInterestsAndFindYourCommunitiesFullstopTextString + "\"; which should be \"Explore your interests and find your communities.\".");

} else {

extentTest.addScreenCaptureFromPath(capturedScreenshotImageFilepathString).pass(MediaEntityBuilder.*createScreenCaptureFromPath*(capturedScreenshotImageFilepathString).build()).log(Status.*FAIL*, "The Amino webpage's tagline text is displayed as the incorrect \"" + exploreYourInterestsAndFindYourCommunitiesFullstopTextString + "\"; which should be \"Explore your interests and find your communities.\".");

}

extentReport.flush();

// Note: Unable to verify any element of pop-up

// Number of valid tagName elements end as per last related non-pop-up WebElement

}

// This method is for the filename for capturing screenshot images which are not in the ExtentReport

public static String takeScreenshot(WebDriver driver) throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

TakesScreenshot screenshot = ((TakesScreenshot) driver);

File sourceFile = screenshot.getScreenshotAs(OutputType.*FILE*);

File destinationFile = new File("C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png");

FileHandler.*copy*(sourceFile, destinationFile);

return "C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + dayFirstDigitChar + daySecondDigitChar + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar + "\_screenshot.png";

}

// This method is for returning the substring of the current LocalDateTome, for the filename of the ExtentReport

public static String currentLocalDateTimeWithDdMmYyFormat() throws IOException {

char dayFirstDigitChar = '0';

char daySecondDigitChar = '0';

char monthFirstDigitChar = '0';

char monthSecondDigitChar = '0';

char yearFirstDigitChar = '0';

char yearSecondDigitChar = '0';

char yearThirdDigitChar = '0';

char yearFourthDigitChar = '0';

char hourFirstDigitChar = '0';

char hourSecondDigitChar = '0';

char minuteFirstDigitChar = '0';

char minuteSecondDigitChar = '0';

char secondFirstDigitChar = '0';

char secondSecondDigitChar = '0';

char subsecondFirstDigitChar = '0';

char subsecondSecondDigitChar = '0';

char subsecondThirdDigitChar = '0';

char subsecondFourthDigitChar = '0';

char subsecondFifthDigitChar = '0';

char subsecondSixthDigitChar = '0';

// Issue - Missing these three digits

char subsecondSeventhDigitChar = '0';

char subsecondEighthDigitChar = '0';

char subsecondNinthDigitChar = '0';

// How to take Selenium screenshot image

LocalDateTime localDateTime = LocalDateTime.*now*();

String initialLocalDateTimeString = localDateTime.toString();

// Will print out

// initial localDateTimeString is "2025-05-09T11:06:10.XXXXXXXXX"

// This has 29 characters (char(s) from 0 to 28)

System.*out*.println("initialLocalDateTimeString is " + "\"" + initialLocalDateTimeString + "\"");

System.*out*.println("initialLocalDateTimeString.length() is " + initialLocalDateTimeString.length() + ".");

// dd

String initialLocalDateTimeStringDaySubstring = initialLocalDateTimeString.substring(8, 10);

// -mm-

String initialLocalDateTimeStringDashMonthDashSubstring = initialLocalDateTimeString.substring(4, 8);

// yyThh:mm:ss.XXXXXXXXX

String initialLocalDateTimeStringYearSubstring = initialLocalDateTimeString.substring(0, 4);

// Despite Java String #StringObject#.substring(X, endIndex being number of characters), meaning (10, 29), an error was displayed, so changed to without endIndex if should reach last character

String initialLocalDateTimeStringTimeSubString = initialLocalDateTimeString.substring(10);

String finalLocalDateTimeString = initialLocalDateTimeStringDaySubstring + initialLocalDateTimeStringDashMonthDashSubstring + initialLocalDateTimeStringYearSubstring + initialLocalDateTimeStringTimeSubString;

System.*out*.println("finalLocalDateTimeString is " + "\"" + finalLocalDateTimeString + "\"");

System.*out*.println("finalLocalDateTimeString.length() is " + finalLocalDateTimeString.length() + ".");

//int testInt=100;

//char testChar = 'a';

//char testChar = 100;

// Able to add int testInt to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testInt + "\_screenshot.png")

// Able to add char to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + testChar + "\_screenshot.png")

// Unable to add String, or char[] (char Array) to new File(C:\\Users\\sohjnthn\\IdeaProjects\\Selenium\_Tutorial\\Selenium\_Screenshot\_Images\\" + finalLocalDateTimeString1 + "\_screenshot.png")

//char[] finalLocalDateTimeString1CharArray = new char[29];

//finalLocalDateTimeString1.getChars(0, 29, finalLocalDateTimeString1CharArray, 0);

if(finalLocalDateTimeString.length()>=1) {

// index 0 character; 1st character

dayFirstDigitChar = finalLocalDateTimeString.charAt(0);

}else{

// index none

System.*out*.println("finalLocalDateTimeString only has 0 characters.");

}

if(finalLocalDateTimeString.length()>=2) {

// index 1 character; 2nd character

daySecondDigitChar = finalLocalDateTimeString.charAt(1);

} else {

// indexes 0 to 1

System.*out*.println("finalLocalDateTimeString only has 1 character.");

}

if(finalLocalDateTimeString.length()>=4) {

// index 3 character; 4th character

monthFirstDigitChar = finalLocalDateTimeString.charAt(3);

} else {

// indexes 0 to 1 or 2

System.*out*.println("finalLocalDateTimeString only has 2 to 3 characters.");

}

if(finalLocalDateTimeString.length()>=5) {

// index 4 character; 5th character

monthSecondDigitChar = finalLocalDateTimeString.charAt(4);

} else {

// index 0 to 3

System.*out*.println("finalLocalDateTimeString only has 4 characters.");

}

if(finalLocalDateTimeString.length()>=7) {

// index 6 character; 7th character

yearFirstDigitChar = finalLocalDateTimeString.charAt(6);

} else {

// indexes 0 to 5

System.*out*.println("finalLocalDateTimeString only has 6 characters.");

}

if(finalLocalDateTimeString.length()>=8) {

// index 7 character; 8th character

yearSecondDigitChar = finalLocalDateTimeString.charAt(7);

} else {

// indexes 0 to 6

System.*out*.println("finalLocalDateTimeString only has 7 characters.");

}

if(finalLocalDateTimeString.length()>=9) {

// index 8 character; 9th character

yearThirdDigitChar = finalLocalDateTimeString.charAt(8);

} else {

// indexes 0 to 7

System.*out*.println("finalLocalDateTimeString only has 8 characters.");

}

if(finalLocalDateTimeString.length()>=10) {

// index 9 character; 10th character

yearFourthDigitChar = finalLocalDateTimeString.charAt(9);

} else {

// indexes 0 to 8; 11th character

System.*out*.println("finalLocalDateTimeString only has 9 characters.");

}

if(finalLocalDateTimeString.length()>=12) {

// index 11 character; 12th character

hourFirstDigitChar = finalLocalDateTimeString.charAt(11);

} else {

// indexes 0 to 10

System.*out*.println("finalLocalDateTimeString only has 11 characters.");

}

if(finalLocalDateTimeString.length()>=13){

// index 12 character; 13th character

hourSecondDigitChar = finalLocalDateTimeString.charAt(12);

} else {

// indexes 0 to 11

System.*out*.println("finalLocalDateTimeString only has 12 characters.");

}

if(finalLocalDateTimeString.length()>=15) {

// index 14 character; 15th character

minuteFirstDigitChar = finalLocalDateTimeString.charAt(14);

} else {

// indexes 0 to 12 or 13

System.*out*.println("finalLocalDateTimeString only has 13 to 14 characters.");

}

if(finalLocalDateTimeString.length()>=16) {

// index 15 character; 16th character

minuteSecondDigitChar = finalLocalDateTimeString.charAt(15);

} else {

// indexes 0 to 14

System.*out*.println("finalLocalDateTimeString only has 15 characters.");

}

if(finalLocalDateTimeString.length()>=18) {

// index 17 character; 18th character

secondFirstDigitChar = finalLocalDateTimeString.charAt(17);

} else {

// indexes 0 to 16

System.*out*.println("finalLocalDateTimeString only has 17 characters.");

}

if(finalLocalDateTimeString.length()>=19) {

// index 18 character; 19th character

secondSecondDigitChar = finalLocalDateTimeString.charAt(18);

} else {

// indexes 0 to 17

System.*out*.println("finalLocalDateTimeString only has 18 characters.");

}

if(finalLocalDateTimeString.length()>=21) {

// index 20 character; 21th character

subsecondFirstDigitChar = finalLocalDateTimeString.charAt(20);

} else {

// indexes 0 to 19

System.*out*.println("finalLocalDateTimeString only has 20 characters.");

}

if(finalLocalDateTimeString.length()>=22) {

// index 21 character; 22nd character

subsecondSecondDigitChar = finalLocalDateTimeString.charAt(21);

} else {

// indexes 0 to 20

System.*out*.println("finalLocalDateTimeString only has 21 characters.");

}

if(finalLocalDateTimeString.length()>=23) {

// index 22 character; 23rd character

subsecondThirdDigitChar = finalLocalDateTimeString.charAt(22);

} else {

// indexes 0 to 21

System.*out*.println("finalLocalDateTimeString only has 22 characters.");

}

if(finalLocalDateTimeString.length()>=24) {

// index 23 character; 24th character

subsecondFourthDigitChar = finalLocalDateTimeString.charAt(23);

} else {

// indexes 0 to 22

System.*out*.println("finalLocalDateTimeString only has 23 characters.");

}

if(finalLocalDateTimeString.length()>=25) {

// index 24 character; 25th character

subsecondFifthDigitChar = finalLocalDateTimeString.charAt(24);

} else {

// indexes 0 to 23

System.*out*.println("finalLocalDateTimeString only has 24 characters.");

}

if(finalLocalDateTimeString.length()>=26) {

// index 25 character; 26th character

subsecondSixthDigitChar = finalLocalDateTimeString.charAt(25);

} else {

// indexes 0 to 24

System.*out*.println("finalLocalDateTimeString only has 25 characters.");

}

if(finalLocalDateTimeString.length()>=27) {

// index 26 character; 27th character

subsecondSeventhDigitChar = finalLocalDateTimeString.charAt(26);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 26 characters.");

}

if(finalLocalDateTimeString.length()>=28) {

// index 27 character; 28th character

subsecondEighthDigitChar = finalLocalDateTimeString.charAt(27);

} else {

// indexes 0 to 25

System.*out*.println("finalLocalDateTimeString only has 27 characters.");

}

if(finalLocalDateTimeString.length()>=29) {

// index 28 character; 29th character

subsecondNinthDigitChar = finalLocalDateTimeString.charAt(28);

} else {

System.*out*.println("finalLocalDateTimeString only has 28 characters.");

}

// There seems to be an error for which the two digits for day are incorrect, so localDateTime.getDayOfmonth is used(), instead of dayFirstDigitChar and daySecondDigitChar

return localDateTime.getDayOfMonth() + "-" + monthFirstDigitChar + monthSecondDigitChar + "-" + yearFirstDigitChar + yearSecondDigitChar + yearThirdDigitChar + yearFourthDigitChar + "T" + hourFirstDigitChar + hourSecondDigitChar + minuteFirstDigitChar + minuteSecondDigitChar + secondFirstDigitChar + secondSecondDigitChar + "." + subsecondFirstDigitChar + subsecondSecondDigitChar + subsecondThirdDigitChar + subsecondFourthDigitChar + subsecondFifthDigitChar + subsecondSixthDigitChar + subsecondSeventhDigitChar + subsecondEighthDigitChar + subsecondNinthDigitChar;

}

}































