TeamMember class Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Description/Purpose | Input | Expected Output | pass/fail |
| 1 | Test booleanTest(), to see if it will properly test if a test case passes or fails. First case will test if a pass, where result and expected match, is recognized | testResult: true  expected: true  (expected is a parameter of the function) | pass | pass |
| 2 | Test booleanTest() to see if it recognizes a fail, result and expected do not match. | testResult: false  expected: true  (expected is a parameter of the function) | fail | pass |
| 3 | Test the constructor using the the toString() method | “Riz”, “2/13/2020” | Riz 2/13/2020 | pass |
| 4 | Test toString() method | TeamMember object with “Tin” , “2/10/2020” | Tin 2/10/2020 | pass |
| 5 | Test equals(): two separate objects but with same name and date; check if object attributes rather than reference is checked | TeamMember “Riz”, “10/29/1998”  TeamMember “Riz”, “10/29/1998” | true | pass |
| 6 | Test equals(): two separate objects with different name and date; check to see if differences in attributes can be detected | TeamMember “Riz”, “2/18/2020”  TeamMember “Tin”,  “2/19/2020” | false | pass |
| 7 | Test equals(): Two TeamMember objects with same name but different dates; will test if date is tested for equality by keeping name constant | TeamMember “Riz”, “2/18/2020”  TeamMember “Riz”,  “2/19/2020” | false | pass |
| 8 | Test equals(): Two TeamMember objects with same date but different names; will test if name is tested for equality by keeping date constant | TeamMember “Riz”, “1/1/2011”  TeamMember “Tin”,  “1/1/2011” | false | pass |
| 9 | Test equals(): One TeamMemeber object and one non-TeamMember object; will test if the method can handle non-TeamMember objects | TeamMember “Riz”, “2/2/2002”  String: “Riz 2/2/2002” | false | pass |
| 10 | Test equals() to see if can handle a null argument passed in | TeamMember  “Tin”, “2/2/2020”  null object | false | pass |
| 11 | Test getStartDate(): test to see if method returns the date that was entered when creating the object; the date will be printed using the Date object’s toString() method | TeamMember “Riz”,  “1/3/2002” | 1/3/2002 | pass |

Date class Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TestCase | Description/Purpose | Input | Expected Output | Status |
| 1 | Test resultVerifier.booleanTest(); check if the result verifier recognizes a passing test case | testResult: true  expected: true | pass | pass |
| 2 | Test if resultVerifier.booleanTest() recognizes a failing test case | testResult: false  expected: true | fail | pass |
| 3 | Test resultVerifier.stringTest(); check if the result verifier recognizes a passing string test case | testResult: “hello”  expected: “hello” | pass | pass |
| 4 | Test resultVerifier.stringTest(); check if the result verifier recognizes a failing string test case | testResult: “testResult”  expected: “expected” | fail | pass |
| 5 | Test Date() constructor; checking to see if object is created, will use toString() to check results | Date(“10/29/1998”) | 10/29/1998 | pass |
| 6 | Test Date(Date d) constructor; check if constructor can build a new Date object when accepting a Date object as argument. Testing will be done with toString() method. | Date(Date object with “10/29/1998”) | 10/29/1998 | pass |
| 7 | Test equals(). Test with two different objects that have the same values for their attributes; tests whether the attribute values are checked instead of just references | Date(“10/29/1998”)  Date(“10/29/1998”) | true | pass |
| 8 | Test equals(). Test with one object that is a Date object and a non-date object; test to see whether equals() can handle objects of different types | Date(“10/29/1998”)  “10/29/1998” (just a string) | false | pass |
| 9 | Test equals(). Test with a null pointer; check if equals() method can handle null arguments | Date(“10/29/1998”)  null pointer | false | pass |
| 10 | Test equals(). Check if equals can handle a Date object that is actually null. | Date(“10/29/1998”)  Date = null | false | pass |
| 11 | Test equals(). Test with two different Date objects with different values for their attributes, in other words different dates. | Date(“10/29/1998”)  Date(“10/15/1998”) | false | pass |
| 12 | Test isValid(). Test if a valid date is evaluated to be valid by the isValid() method. | Date(“10/29/1998”) | true | pass |
| 13 | Test isValid(). Test with an invalid date that has month greater than 12; check if isValid() can detect invalid month in a Date object. | Date(“12/21/1999”) | false | pass |
| 14 | Test isValid(). Test with an invalid date that has day 31 for a 30 day month; checks if isValid() can detect if the number of days and the month is consistent. | Date(“4/31/2021”) | false | pass |
| 15 | Test isValid(). Test with an invalid date that has day 29 for a year that is not a leap-year. Checks if isValid() can accurately detect if a year is a leap year and whether it has an accurate number of days. | Date(“2/29/2019”) | false | pass |
| 16 | Test isValid(). Test with a valid date that is a leap year date. Checks if isValid() can detect an accurate leap year date. | Date(“2/29/2020”) | true | pass |
| 17 | Test isLeapYear(). Test with a year that is not a leap year to see if this method can detect non-leap years. | 1900 | false | pass |
| 18 | Test isLeapYear(). Test with a year that is a leap year; check if method can detect a proper leap year | 2020 | true | pass |
| 19 | Test toString(). Test with a Date object to see if the toString() method will return the date in proper string format. | Date(“2/10/2020”) | 2/10/2020 | pass |