Assertions in Ballerina

Analysis on current AssertEquals implementation

Value equality vs. reference equality

== for deep value equality of anydata|error typed values.
=== for reference equality.

Types and how they are handled when asserting

- Anydata compatible types - pure values(except error)

()
Boolean
Int
Float
Decimal
String
Byte

Straightforward value comparison

Tuple

```
[string|int, float, boolean] t1 = [1, 1.0, false];
[int, float|string, boolean] t2 = [1, 1.0, false];
```

assertEquals(t1,t2) → true

Type along with the values are being checked.

```
[string|int, decimal, boolean] t1 = [1, 1.0, false];
[int, float|string, boolean] t2 = [1, 1.0, false];
```

assertEquals(t1,t2) → Assertion Failed!

(anydata|error)[]

```
float a = 1.1;
decimal b = 1.1;
(anydata|error)[] arr1 = ["wwjdsd","dds",a];
(anydata|error)[] arr2 = ["wwjdsd","dds",b];
```

assertEquals(arr1,arr2) → Assertion Failed!

map<anydata|error>

```
map<string> testMap1 = {code:"hello", action: "print"};
map<string> testMap2 = {code1:"hello", action: "print"};
```

assertEquals(testMap1,testMap2) → Assertion Failed!

```
Key and values both are compared.
```

Xml

a sequence of zero or more XML items. Each item can be an element, a text, a comment, or a processing instruction.

```
// The XML element. There can be only one root element.
xml x1 = xml `<book>The Lost World</book>`;
io:println(x1);

// The XML text.
xml x2 = xml `Hello, world!`;
io:println(x2);

// The XML comment.
xml x3 = xml `<!--I am a comment-->`;
io:println(x3);

// The XML processing instructions.
xml x4 = xml `<?target data?>`;
io:println(x4);
```

```
// Multiple XML items can be combined to form a sequence of XML. The
resulting sequence is another XML on its own.
xml x5 = x1 + x2 + x3 + x4;

xml x6= xml `<book>The Lost World</book>` + xml `Hello, world!`+ xml `<!--I
am a comment-->`+ xml `<?target data?>`;
```

assertEquals(x5,x6) → true

```
xml x5 = x1 + x2 + x3 + x4;
xml x6= xml `<book>The Lost World</book>` + xml `Hello, world!`+xml `<!--I am
a comment-->`+ "<?target data?>";
```

assertEquals(x5,x6) → Assertion Failed!

Table

Assertion checks are failing. - ballerina 1.2.6
Passes in Swanlake with the new readonly option.

```
table<Employee> tbEmployee = table {
    {key id, name, salary},
    [
        {1, "Mary", 300.5},
        {2, "John", 200.5},
        {3, "Jim", 330.5}
    ]
};
table<Employee> tbEmployeeTest = table {
    {key id, name, salary},
    [
        {1, "Mary", 300.5},
        {2, "John", 200.5},
        {3, "Jim", 330.5}
    ]
};
```

assertEquals(tbEmployeeTest, tbEmployee);

Tries to do a value comparison but fails.

Json

checked by traversing through the json objects' fields

```
   json a = {"name": "Ballerina"};
   json b = {"name": "Ballerina"};
```

assertEquals(a, b) → true

Record with type of each individual field is a subtype of anydata

```
type Student record {
   string name;
   int age;
   Grades grades;
};

Student anne = {
   name: "Anne",
   age: 18,
   grades: {
       maths: 70,
       physics: 80,
       chemistry: 55
   },
   ...address
};

Student anne1 = {
   name: "Anne",
   age: 18,
   grades: {
       maths: 70,
       physics: 80,
       chemistry: 55
   },
   "city": "Colombo",
   "country": "Sri Lanka"
};
```

test:assertEquals(anne, anne1) → true

Does a value comparison

- Other data types under Any type

Object

```
type Person object {
   public string name = "";
   public int age = 0;
   public Person? parent = ();
   private string email = "default@abc.com";
   string address = "No 20, Palm grove";
};

Person p1 = new;
Person p2 = new ();
```

test: assertEquals(p1,p2); → false

This fails because the objects are checked based on reference. We need to have a way to check the deep value equality for objects.

Object[]

```
type Person object {
   public string name = "";
   public int age = 0;
   public Person? parent = ();
   private string email = "default@abc.com";
   string address = "No 20, Palm grove";
};
Person[]  employees = [p1,p2];
```

test: assertEquals(employees, [p1,p2]); → false

```
Person p1=new Person();
Person p2=new Person();
Person[] emp = [p1,p2];
Person[] emp2 = [p1,p2];
```

assertEquals(emp, emp2);→ false

This fails because the array equality is checked not based on the elements, but based on the reference equality of the array itself.

Record with one or more fields being object type

```
type EnrolledStudent record {
   int id;
   Person person;
};

EnrolledStudent student1 = {
```

```
  id: 1,
  person: p1
};

EnrolledStudent student2 = {
  id: 1,
  person: p1
};
```

AssertEquals(student1,student2) → false

This needs to be fixed to do a deep value equality.

Plan for Assertions in Ballerina 1.2.x

- Update the how the current implementation works in docs.
- Get table equality comparison fixed based on [1]