Generics

Agenda

Generic classes

Generic constraints

Generic methods

Generic interfaces

Generic classes

Generic is what is shared by a whole group of similar things

https://dictionary.cambridge.org/

Generics

It’s a type that has type parameters

These parameters act like placeholders that can hold any type

The type parameters can be defined on the entire type or on one or more members (fields, properties, methods, delegates)

Generic class declaration

```
public class MyGenericClass<T>
{
    T reference;
    T MyMethod(T parameter)
    {
        return reference;
    }
}
```

Generic classes

Has one or more parameter types on it

The actual parameter type is defined when the generic type is created

It can’t be instantiated without a concrete type

Generic class declaration

```
public class MyGenericClass<T,U>
{
    T MyMethod1(U parameter)
    {
        //
    }
    U MyMethod2(T parameter)
    {
        //
    }
}
```

Generic class instantiation

```
var myGenericTypeInstance = new MyGenericClass<int, string>();
```

Generic class instantiation

```
var myGenericTypeInstance = new MyGenericClass<Apple,Banana>();
```

Demo

Create generic classes

Create generic class instances

Type constraints

Constraints

Constraints can be defined on class type parameters

Constraints

```
public class MyGenericClass<T> where T : class
{
    T reference;
    T MyGenericMethod()
    {
        return reference;
    }
}
```

Constraints

```
var myGenericTypeInstance = new MyGenericClass<Banana>();
```

✓

Constraints

```
var myGenericTypeInstance = new MyGenericClass<int>();
```

Demo

Create generic type constraints

Generic methods

Generic methods

Has a type parameters on one of its type parameters or on its return type

Generic methods

```
public class MyClass<T>
{
    public T MyGenericMethod()
    {
        // implementation
    }
}
```

Generic methods

Can be overloaded with different type parameters

The type parameter can be defined in the method declaration only

Generic methods on non generic class

```
public class MyClass
{
    public void MyGenericMethod<T>()
    {
        //
    }

    public void MyGenericMethod<T,U>()
    {
        //
    }
}
```

Generic method constraint

```
public class MyClass
{
    public T MyGenericMethod<T>() where T : class, new()
    {
        return new T();
    }
}
```

Demo

Create generic methods

Create generic method constraints

Generic interfaces

Generic interfaces

Interfaces can be generic

Like concrete types, the type parameters can be set on the interface or on its methods

A generic class or interface can implement a generic base interface

Generic interfaces

```
public interface IMyGenericInterface<T> where T : class
{
    T MyMethod();
}
```

Demo

Create and implement generic interface

Generics benefits

Benefits

Type safety

Code reuse

Better performance

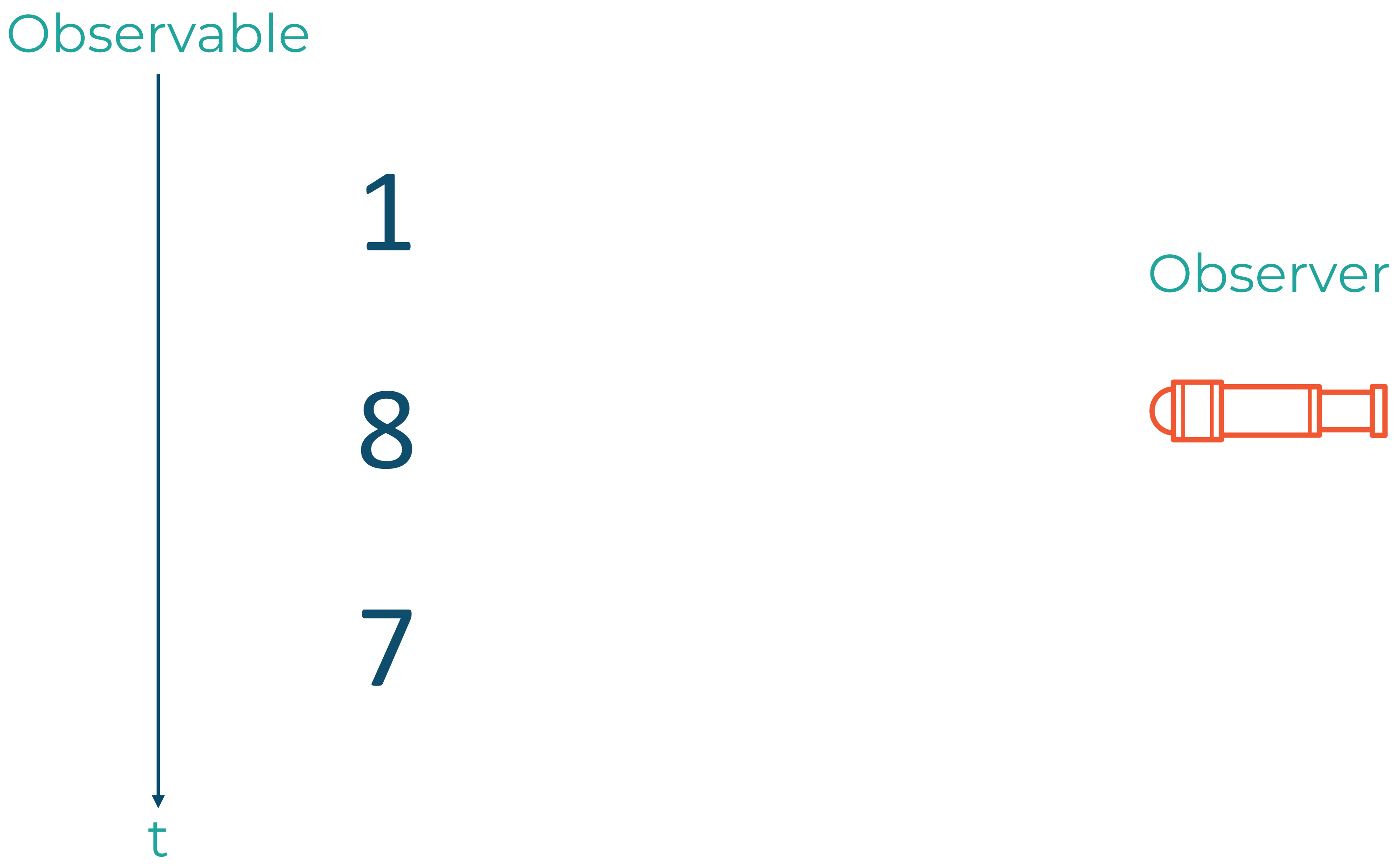
Challenge

Implement the observer pattern

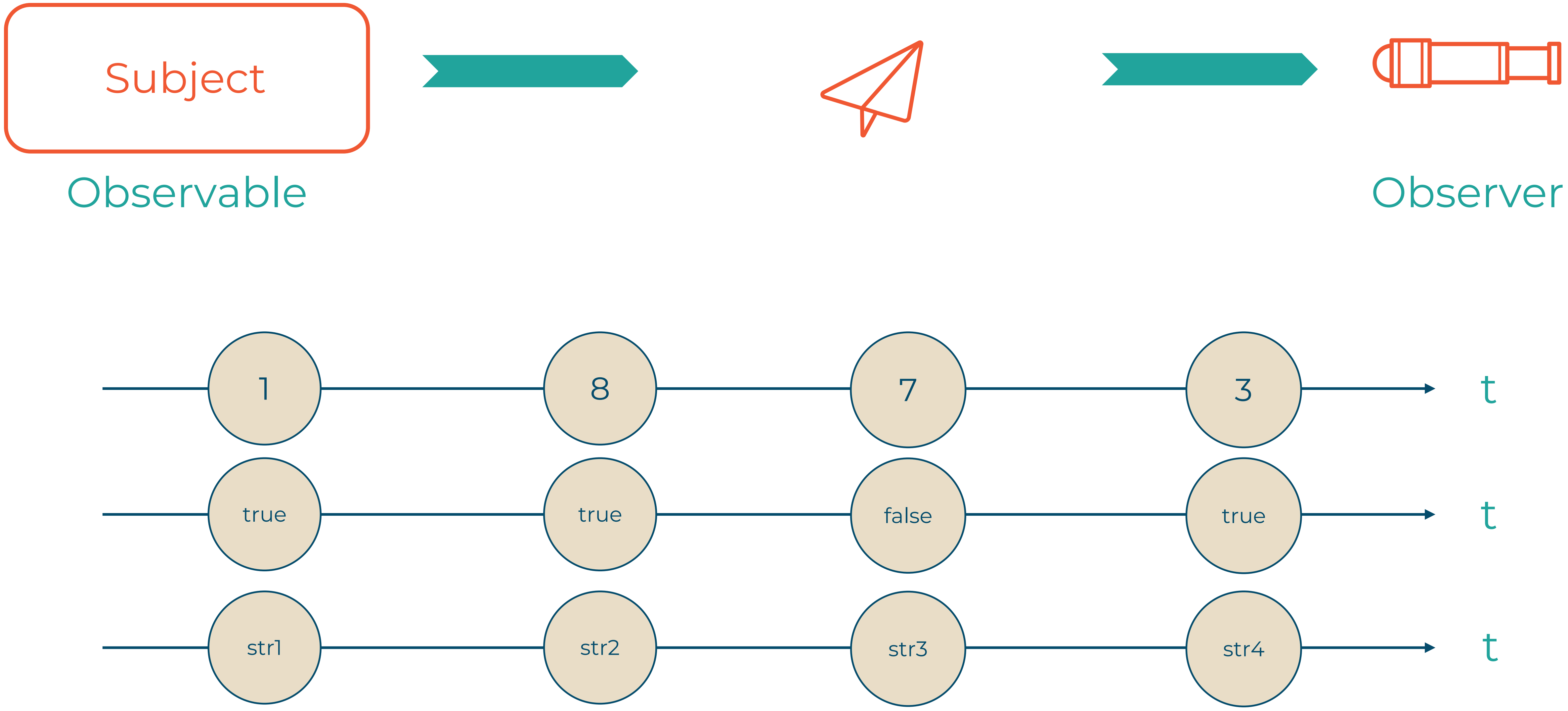
An observer vs an observable

The observer receives notifications from the observable

Observable/Observer pattern



Observable/Observer



Requirements

Create an Observer/Observable class

Simplified implementation, just a Notify method

The types must be generic

Requirements

The observable type contains the object to observe, the generic subject

Add a type constraint, only for value types

The subscribe/unsubscribe methods allow the observer to subscribe to notifications and unsubscribe

Summary

A generic type is a type that is typical of one or more other types

A generic class has one or more type parameters

The related types are specified in the instantiation of the generic class

A generic method can be defined on generic and non generic classes

An interface can be generic

You can define type constraints on generic classes, methods and interfaces