Module 4: Interfaces for Abstraction
Topic 2.2: Type Assertions

Concealing Type Differences

- Interfaces hide the differences between types

```
func FitInYard(s Shape2D) bool {
     if (s.Area() < 100 &&
         s.Perimeter() < 100) {
          return true
     }
     return false
}
```

- Sometimes you need to treat different types in different ways

Exposing Type Differences

- Example: Graphics program
- **`DrawShape()`** will draw any shape
  - `func DrawShape(s Shape2D) { …`
- Underlying API has different drawing functions for each shape
  - `func DrawRect(r Rectangle) { …`
  - `func DrawTriangle(t Triangle) { …`
- Concrete type of shape s must be determined

Type Assertions

- Type assertions can be used to determine and extract the underlying concrete type

```
func DrawShape(s Shape2D) bool {
      rect, ok := s.(Rectangle)
```

- Type assertion extracts Rectangle from Shape2D
  - Concrete type in parentheses
- If interface contains concrete type
  - rect == concrete type, ok == true
- If interface does not contain concrete type
  - rect == zero, ok == false

Type Assertions for Disambiguation

```
func DrawShape(s Shape2D) bool {
     rect, ok := s.(Rectangle)
     if ok {
          DrawRect(rect)
     }
     tri, ok := s.(Triangle)
     if ok {
          DrawTriangle(tri)
     }
}
```

Type Switch

- Switch statement used with a type assertion

```
func DrawShape(s Shape2D) bool {
      switch sh := s.(type) {
      case Rectangle:
            DrawRect(sh)
      case Triangle:
            DrawTriangle(sh)
      }
}
```