Go Documentation Server

Package booking

```
import
"TanJunJie_GoInAction2Submission/TanJunJie_GoInAction2Submission/booking"
```


Overview

Package booking contains the data structures and algorithms used to implement an appointment booking system.

Index

Package files

appointment.go availability.go doc.go doctors.go patients.go

Constants

These define a doctor's default availability. Consultation hours are from 09:00-16:00. No consultation available during lunch time, 12:00. Available dates start from one week from current date to the week after.

```
const (
    workStartHour   int = 9
    workEndHour     int = 16
    oneWeekFromNow  int = 7
    twoWeeksFromNow int = 15
    lunchHour       int = 12
)
```

Variables

Error codes returned by failures to find a match or invalid inputs.

```
var (
    ErrEmptyQueue      = errors.New("Appointments List is empty. There are no
appointments available.")
    ErrInvalidAppt     = errors.New("Invalid Appointment")
    ErrInvalidName     = errors.New("Invalid Name")
    ErrInvalidList     = errors.New("List is empty!")
    ErrDocNotFound     = errors.New("The named doctor was not found!")
    ErrPatientNotFound = errors.New("The named patient was not found!")

    patientLock, availabilityLock, appointmentLock, updateAppointmentLock
sync.Mutex
)
```

func **Insert**

```
func Insert(arr []time.Time, t time.Time) []time.Time
```

Insert adds a time slot to a slice of time slots in sorted order.

func **initAvailability**

```
func initAvailability() []time.Time
```

initAvailability returns a default slice of available time slots.

type **Appointment**

Appointment contains information regarding the appointment.

```
type Appointment struct {
    dateTime    time.Time
    patientName string
    doctorName  string
}
```

type **Doctor**

Doctor is a Node in the Doctors BST.

```
type Doctor struct {
    Info  DoctorInfo
    left  *Doctor
    right *Doctor
}
```

func (*Doctor) AddAvailability

```
func (d *Doctor) AddAvailability(dt time.Time) error
```

AddAvailability frees up a time slot in a doctor's schedule.

func (*Doctor) IsAvailableAt

```
func (d *Doctor) IsAvailableAt(dt time.Time) bool
```

IsAvailableAt shows the availability of a doctor on a particular date and time.

func (*Doctor) IsAvailableOn

```
func (d *Doctor) IsAvailableOn(dt time.Time) bool
```

IsAvailableOn shows the availability of a doctor on a particular day.

func (*Doctor) RemoveAvailability

```
func (d *Doctor) RemoveAvailability(dt time.Time) error
```

RemoveAvailability removes a time slot in a doctor's schedule.

func (*Doctor) ShowAvailability

```
func (d *Doctor) ShowAvailability() string
```

ShowAvailability shows all timeslots for which a doctor is available.

type DoctorInfo

DoctorInfo contains information regarding a doctor.

```
type DoctorInfo struct {
    DoctorName   string
    Appointments Queue
    Availability []time.Time
}
```

type Doctors

Doctors is a Binary Search Tree.

```
type Doctors struct {
    root *Doctor
}
```

func CreateDoctors

```
func CreateDoctors() Doctors
```

CreateDoctors returns an empty Doctors BST.

func (*Doctors) InitDoctors

```
func (d *Doctors) InitDoctors(list []string) error
```

InitDoctors iterates through a list of doctor names to add Doctor nodes to the Doctors BST.

func (*Doctors) Print

```
func (d *Doctors) Print() string
```

Print returns a formatted string that contains the names of all the doctors in the Doctors BST.

func (*Doctors) SearchDoctor

```
func (d *Doctors) SearchDoctor(name string) (*Doctor, error)
```

SearchDoctor is a convenience function that wraps searchDocNodes, requiring only a doctor's name to be passed to it.

func (*Doctors) ShowAllAvailability

```
func (d *Doctors) ShowAllAvailability(dt time.Time) string
```

ShowAllAvailability returns a string of doctors that are available on a particular date.

func (*Doctors) add

```
func (d *Doctors) add(name string) error
```

add is a convenience function that wraps addDoctor, requiring only that a new doctor's name is passed to it.

func (*Doctors) addDoctor

```
func (d *Doctors) addDoctor(doc **Doctor, name string) error
```

addDoctor creates a Doctor node and adds it to the Doctors BST.

func (*Doctors) printDoctors

```
func (d *Doctors) printDoctors(doc *Doctor) string
```

printDoctors does an in-order traversal through the BST and returns the doctor's name.

func (*Doctors) searchDocNodes

```
func (d *Doctors) searchDocNodes(doc *Doctor, name string) (*Doctor, error)
```

searchDocNodes recursively searches the Doctors BST for a node with the specified name.

func (*Doctors) showAvailabilityOn

```
func (d *Doctors) showAvailabilityOn(doc *Doctor, dt time.Time) string
```

showAvailabilityOn recursively checks if a doctor is available on a particular date.

type **Patient**

Patient is a Node in the Patients BST.

```
type Patient struct {
    Info  PatientInfo
    left  *Patient
    right *Patient
}
```

type **PatientInfo**

PatientInfo contains information regarding a patient.

```
type PatientInfo struct {
    PatientName        string
    AppointmentHistory Queue
}
```

type **Patients**

Patients is a Binary Search Tree.

```
type Patients struct {
    root *Patient
}
```

func CreatePatients

```
func CreatePatients() Patients
```

CreatePatients returns an empty Patients BST.

func (*Patients) Add

```
func (p *Patients) Add(PatientName, doctorName string, apptdt time.Time) error
```

Add inserts a Patient node to the Patients BST.

func (*Patients) Print

```
func (p *Patients) Print() string
```

Print returns a formatted string that contains the names of all the patients in the Patients BST.

func (*Patients) SearchPatient

```
func (p *Patients) SearchPatient(name string) (*Patient, error)
```

SearchPatient is a convenience function that wraps searchPatNodes, requiring only a doctor's name to be passed to it.

func (*Patients) addPatient

```
func (p *Patients) addPatient(pat **Patient, PatientName, doctorName string,
apptdt time.Time) error
```

addPatient creates a Patient node and adds it to the Patients BST.

func (*Patients) printPatients

```
func (p *Patients) printPatients(pat *Patient) string
```

printPatients does an in-order traversal through the BST and returns the patients's name.

func (*Patients) searchPatNodes

```
func (p *Patients) searchPatNodes(pat *Patient, name string) (*Patient, error)
```

searchPatNodes recursively searches the Patients BST for a node with the specified name.

type Queue

```
type Queue struct {
    front *QueueNode
    back  *QueueNode
    size  int
}
```

func (*Queue) AddAppointment

```
func (q *Queue) AddAppointment(dt time.Time, patientName, doctorName string)
error
```

AddAppointment inserts an appointment into the queue.

func (*Queue) FinishAppointment

```
func (q *Queue) FinishAppointment() (Appointment, error)
```

FinishAppointment deletes the first appointment from the queue.

func (*Queue) PrintAllNodes

```
func (q *Queue) PrintAllNodes() (string, error)
```

PrintAllNodes returns a formatted string of all the appointments in the queue.

func (*Queue) RemoveAppointment

```
func (q *Queue) RemoveAppointment(dt time.Time, patientName string) error
```

RemoveAppointment removes a specific appointment from the queue.

func (*Queue) SearchAppointment

```
func (q *Queue) SearchAppointment(dt time.Time, patientName string)
(Appointment, error)
```

SearchAppointment searches the queue for the patient's appointment at the specified time.

func (*Queue) isEmpty

```
func (q *Queue) isEmpty() bool
```

func (*Queue) updateAppointment

```
func (q *Queue) updateAppointment(oldappt, newappt Appointment) error
```

updateAppointment updates the details of an appointment.

type QueueNode

```
type QueueNode struct {
    appointment Appointment
    next        *QueueNode
}
```

func (*QueueNode) Print

```
func (q *QueueNode) Print() string
```

Print returns a formatted string of an appointment.

Subdirectories

Name

..

helperFunctions