Solutions for UML Class Diagrams

Chapter 9

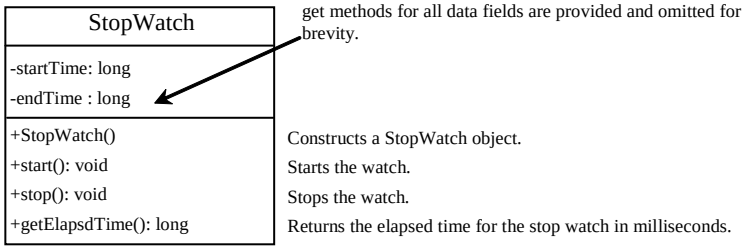
Exercise 9.1: Rectangle

Rectangle	
width: double	The width of this rectangle (default 1).
height: double	The height of this rectangle (default 1).
Rectangle()	Constructs a default rectangle.
Rectangle(width: double, height: double)	Constructs a rectangle with the specified width and height.
getArea(): double	Returns the area of this rectangle.
getPerimeter(): double	Returns the perimeter of this rectangle.

Exercise 9.2: Stock

Stock	
symbol: String	The symbol of this stock.
name: String	The name of this stock.
previousClosingPrice: double	The previous closing price of this stock.
currentPrice: double	The current price of this stock.
Stock(symbol: String, name: String)	Constructs a stock with a specified symbol and a name.
getChangePercent(): double	Returns the percentage of change of this stock.

Exercise 9.6: StopWatch



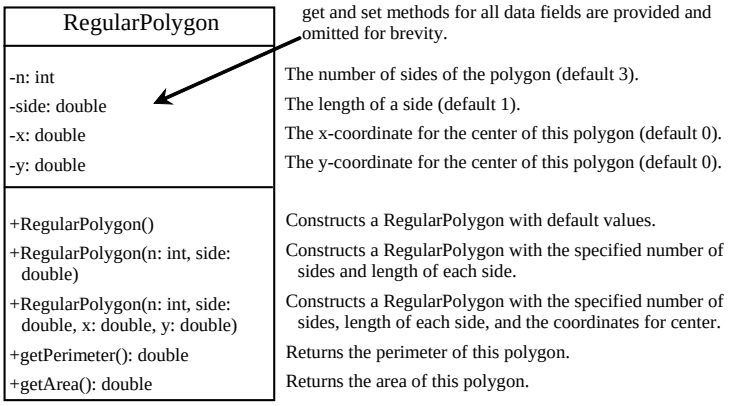
Exercise 9.7: Account

Account	
-id: int	The ID of this account (default 0).
-balance: double	The balance of this account (default 0).
-annualInterestRate: double	The annual interest rate of this account (default 0).
-dateCreated: java.util.Date	The date when this account was created.
+Account()	Constructs a default account.
+Account(id: int, balance: double)	Constructs an account with the specified ID and balance.
+getId(): int	Returns the ID of this account.
+getBalance(): double	Returns the balance of this account.
+getAnnualInterestRate(): double	Returns the interest rate of this account
+getMonthlyInterestRate(): double	Returns the monthly interest rate of this account
+getDateCreated(): java.util.Date	Returns the date when this account was created.
+setId(id: int): void	Sets a new ID of this account.
+setBalance(balance: double): void	Sets a new balance for this account.
+setAnnualInterestRate(annualInterestRate: double): void	Sets a new interest rate for this account.
+getMonthlyInterest(): double	Returns the monthly interest of this account.
+withdraw(amount: double): void	Withdraws the specified amount from this account.
+deposit(amount: double): void	Deposits the specified amount to this account.

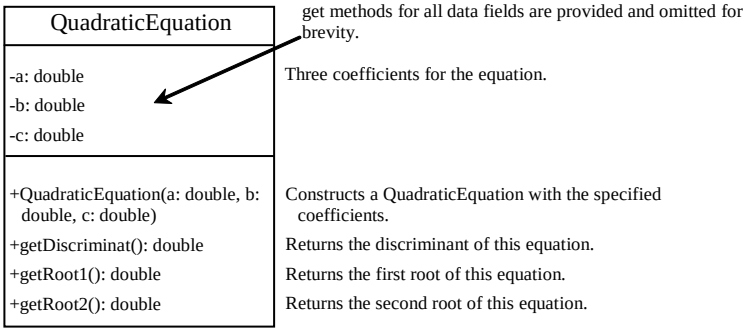
Exercise 9.8: Fan

Fan	
+SLOW = 1	Constant.
+MEDIUM = 2	Constant.
+FAST = 3	Constant.
-speed: int	The speed of this fan (default 1).
-on: boolean	Indicates whether the fan is on (default false).
-radius: double	The radius of this fan (default 5).
-color: String	The color of this fan (default white).
+Fan()	Constructs a fan with default values.
+getSpeed(): int	Returns the speed of this fan.
+setSpeed(speed: int): void	Sets a new speed for this fan.
+isOn(): boolean	Returns true if this fan is on.
+setOn(on: boolean): void	Sets this fan on to true or false.
+getRadius(): double	Returns the radius of this fan.
+setRadius(radius: double): void	Sets a new radius for this fan.
+getColor(): String	Returns the color of this fan.
+setColor(color: String): void	Sets a new color for this fan.
+toString(): String	Returns a string representation for this fan.

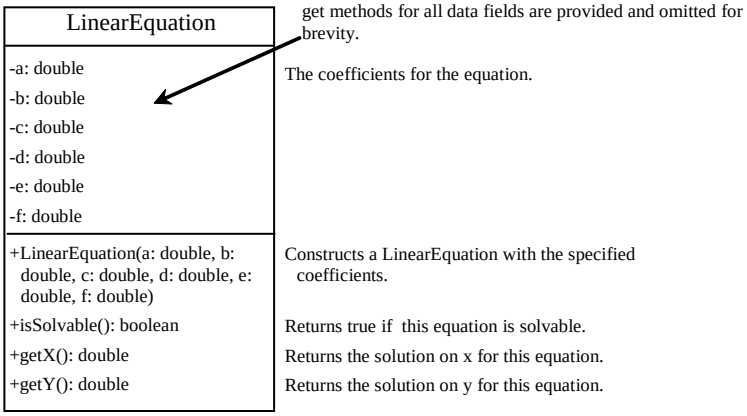
Exercise 9.9: RegularPolygon



Exercise 9.10: QuadraticEquation



Exercise 9.11: LinearEquation



Chapter 10

Exercise 10.1: Time

Time	
-hour: int	The hour for the time.
-minute: int	The minute for the time.
-second: int	The second for the time.
+Time()	Constructs Time for the current time.
+Time(elapseTime: long)	Constructs Time with a specified elapse time in milliseconds.
+Time(hour: int, minute: int, second: int)	Constructs Time with the specified hour, minute, and second.
+getHour(): int	Returns the clock hour for the time.
+getMinute(): int	Returns the minute for the time.
+getSecond(): int	Returns the second for the time.
+setTime(elapsedTime: long): void	Sets a time for the specified elapsed time.

Exercise 10.3: MyInteger

MyInteger	
-value: int	An int value for the object.
+MyInteger(value: int)	Constructs a MyInteger object with the specified int value.
+getValue(): int	Returns the value in this object.
+isPrime(): boolean	Returns true if the value in this object is prime.
+isPrime(value: int): boolean	Returns true if a specified int value is prime.
+isPrime(value: MyInteger): boolean	Returns true if the value in a specified MyInteger object is prime.
+isEven(): boolean	Returns true if the value in this object is even.
+isEven(value: int): boolean	Returns true if a specified int value is even.
+isEven(value: MyInteger): boolean	Returns true if the value in a specified MyInteger object is even.
+isOdd(): boolean	Returns true if the value in this object is odd.
+isOdd (value: int): boolean	Returns true if a specified int value is odd.
+isOdd(value: MyInteger): boolean	Returns true if the value in a specified MyInteger object is odd.
+equals(anotherValue: int): boolean	Returns true if a specified int value is equal to the value in this object.
+equals(anotherValue: MyInteger): boolean	Returns true if the value in a specified MyInteger object is equal to the value in this object.
+parseInt(value: String): int	Returns the int value for the specified string.

Exercise 10.4: MyPoint

MyPoint	
-x: double	x-coordinate of this point.
-y: double	y-coordinate of this point.
+MyPoint()	Constructs a Point object at (0, 0).
+MyPoint(x: double, y: double)	Constructs an object with specified x and y values.
+getX(): double	Returns x value in this object.
+getY(): double	Returns y value in this object.
+distance(secondPoint: MyPoint): double	Returns the distance from this point to another point.
+distance(p1: Point, p2: MyPoint): double	Returns the distance between two points.

Exercise 10.8: Tax

Tax
-filingStatus: int -brackets: int[][] -rates: double[] -taxableIncome: double
+Tax() +Tax(filingStatus: int, brackets: int[], rates: double[], taxableIncome: double) +getFilingStatus(): int +setFilingStatus(filingStatus: int): void +getBrackets(): int[][] +setBrackets(brackets: int[][]): void +getRates(): double[] +setRates(rates: double[]): void +getTaxableIncom(): double +setTaxableIncome(taxableIncome: double): void +getTax(): double

Exercise 10.11: Circle2D

Circle2D
-x: double -y: double -radius: double
+Circle2D() +Circle2D(x: double, y: double, radius: double) +getX(): double +getY(): double +setX(x: double): void +setY(y: double): void +getRadius(): double +setRadius(radius: double): void +getPerimeter(): double +getArea(): double +contains(x: double, y: double): boolean +contains(circle: Circle2D): boolean +overlaps(circle: Circle2D): boolean

Exercise 10.12: Triangle2D

Triangle2D
-p1: MyPoint -p2: MyPoint -p3: MyPoint
+Triangle2D() +Triangle2D(x1: double, y1: double, x2: double, y2: double, x3: double, y3: double,) +Triangle2D(p1: MyPoint, p2: MyPoint, p3: MyPoint) +getP1(): MyPoint +setP1(p1: MyPoint): void +getP2(): MyPoint +setP2(p2: MyPoint): void +getP3(): MyPoint +setP3(p3: MyPoint): void +getPerimeter(): double +getArea(): double +contains(p: MyPoint): boolean +contains(t: Triangle2D): boolean +overlaps(t: Triangle2D): boolean

Exercise 10.13: MyRectangle2D

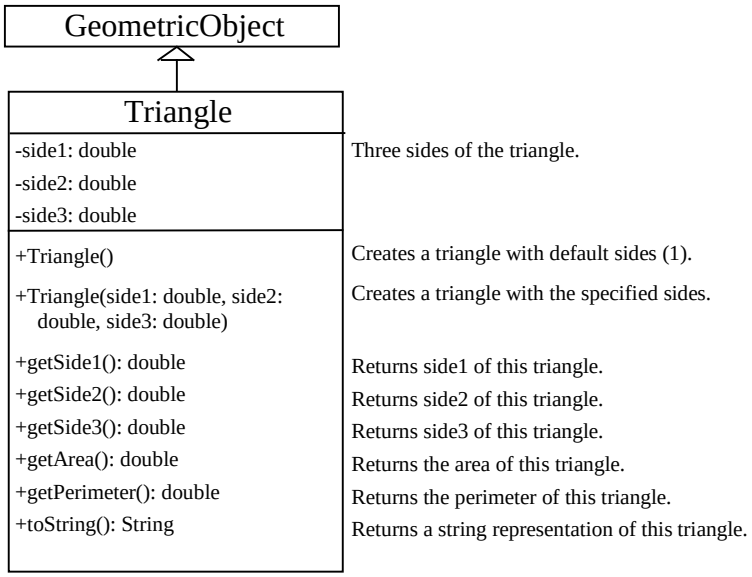
MyRectangle2D
-x: double -y: double -width: double -height: double
+MyRectangle2D() +MyRectangle2D(x: double, y: double, width: double, height: double) +getX(): double +setX(x: double): void +getY():double +setY(y: double): void +getWidth(): double +setWidth(width: double): void +getHeight(): double +setHeight(height: double): void +getRadius(): double +getPerimeter(): double +getArea(): double +contains(x: double, y: double): boolean +contains(r: Rectangle2D): boolean +overlaps(r: Rectangle2D): boolean

Exercise 10.14: MyDate

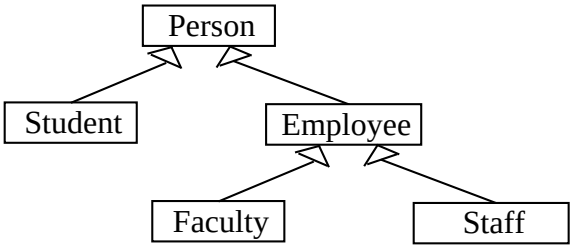
MyDate	
-year: int	The year for the date.
-month: int	The month for the date.
-day: int	The day for the date.
+MyDate()	Constructs MyDate for the current date.
+MyDate(elapsedTime: long)	Constructs MyDate with a specified elapse time in milliseconds.
+getYear(): int	Returns the year for the date.
+getMonth(): int	Returns the month for the date.
+getDay(): int	Returns the day for the date.
+setDate(elapsedTime: long): void	Sets a new date using the elapsed time.

Chapter 11

Exercise 11.1: Triangle



Exercise 11.2: Person, Student, Staff, Employee

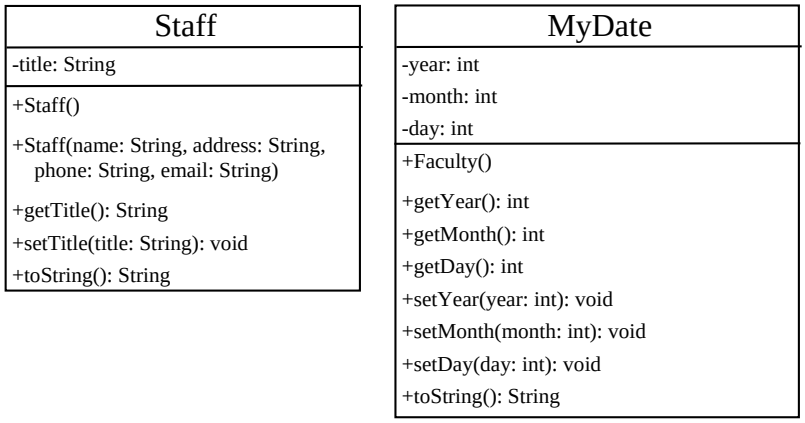


Person
-name: String -address: String -phone: String -email: String
+Person() +Person(name: String, address: String, phone: String, email: String) +getName(): String +getAddress(): String +getPhone(): String +getEmail(): String +setName(name: String): void +setAddress(address: String): void +setPhone(phonee: String): void +setEmail(email: String): void +toString(): String

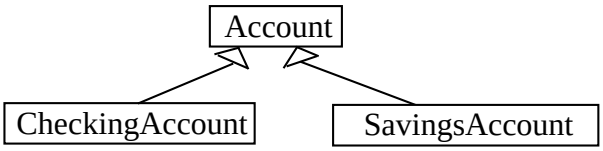
Student
-status: String
+Student() +Student(name: String, address: String, phone: String, email: String) +getStatus(): String +setStatus(status: String): void +toString(): String

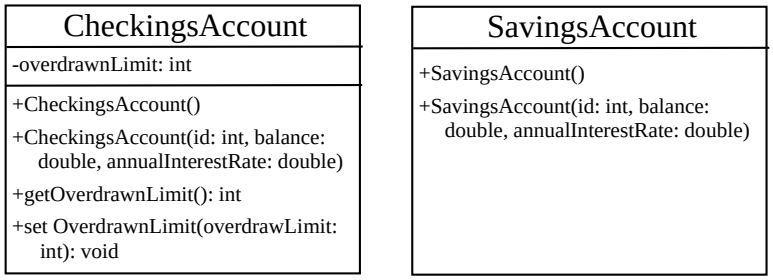
Employee
-office: String -salary: int -dateHired: java.util.Date
+Employee() +Employee(name: String, address: String, phone: String, email: String) +getOffice(): String +getSalary(): int +getDateHired(): Date +setOffice(office: String): void +setSalary(salary: int): void +setDateHired(dataHired: Date): void +toString(): String

Faculty
-officeHour: String -rank: String
+Faculty() +Faculty(name: String, address: String, phone: String, email: String) +getOfficeHour(): String +setOfficeHour(officeHour: String): void +getRank(): String +setRank(rank: String): void +toString(): String



Exercise 11.3: Account

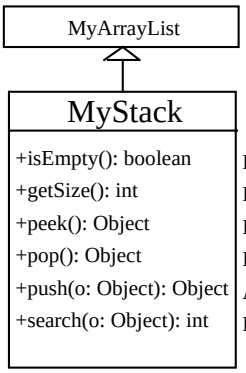




Exercise 11.5: Course

Course	
`-courseName: String`	The name of the course.
`-students: ArrayList<String>`	An ArrayList to store the students for the course.
`+Course(courseName: String)`	Creates a course with the specified name.
`+getCourseName(): String`	Returns the course name.
`+addStudent(student: String): void`	Adds a new student to the course.
`+dropStudent(student: String): void`	Drops a student from the course.
`+getStudents(): String[]`	Returns the students in the course.
`+getNumberOfStudents(): int`	Returns the number of students in the course.

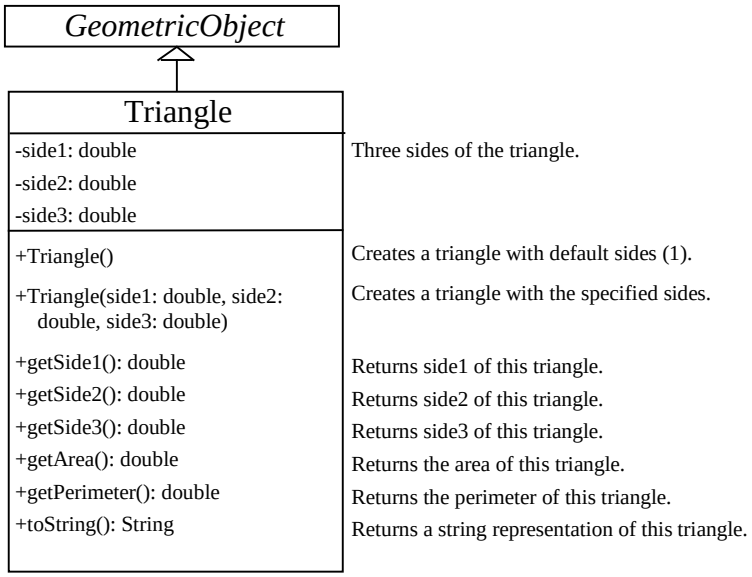
Exercise 11.10: MyStack



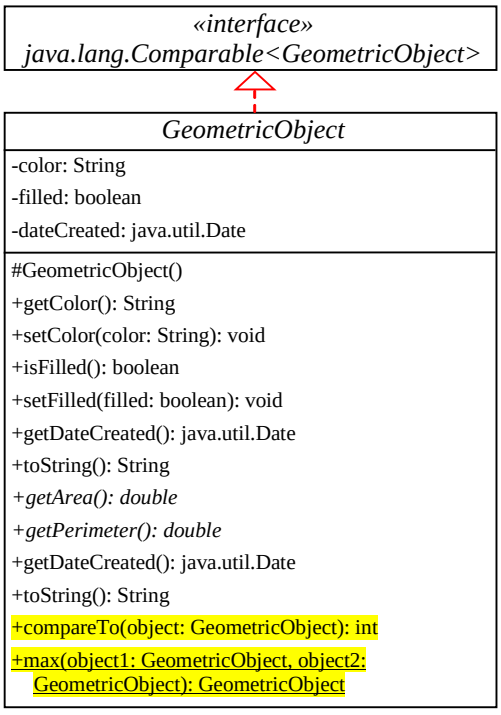
Returns true if this stack is empty.
Returns the number of elements in this stack.
Returns the top element in this stack.
Returns and removes the top element in this stack.
Adds a new element to the top of this stack.
Returns the position of the first element in the stack from the top that matches the specified element.

Chapter 13

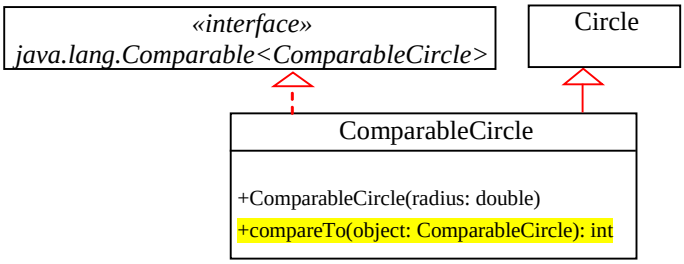
Exercise 13.1: Triangle



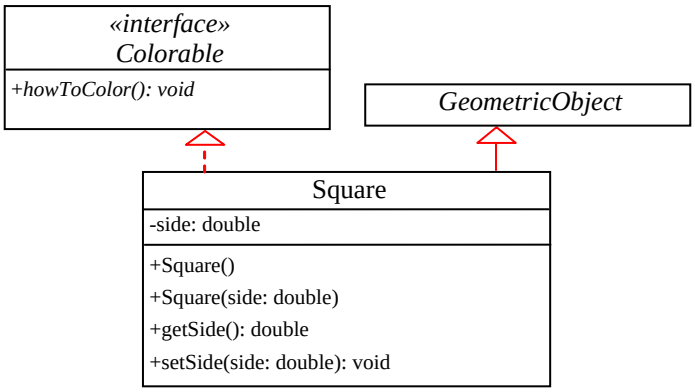
Exercise 13.5: GeometricObject



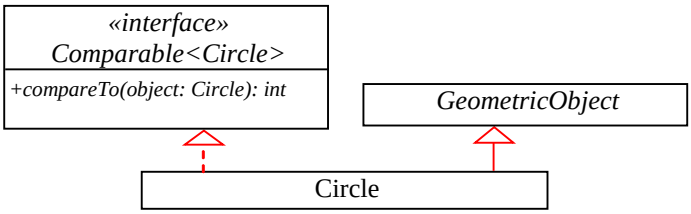
Exercise 13.6: ComparableCircle


«interface»
java.lang.Comparable<ComparableCircle>
Circle
ComparableCircle
+ComparableCircle(radius: double)
+compareTo(object: ComparableCircle): int

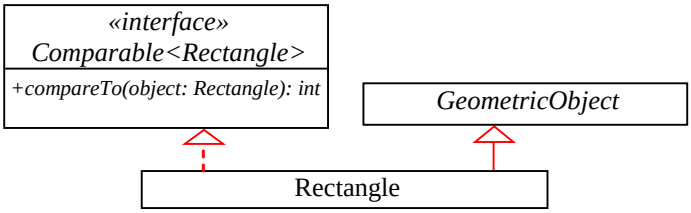
Exercise 13.7: Colorable, Square



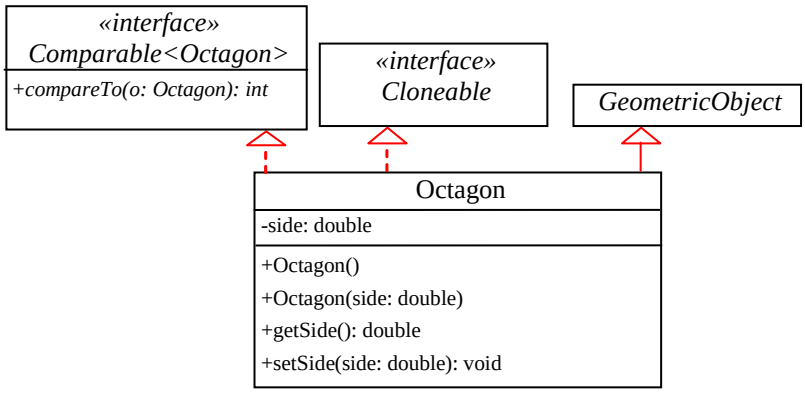
Exercise 13.9: Circle



Exercise 13.10: Rectangle



Exercise 13.11: Octagon



Exercise 13.17: Complex