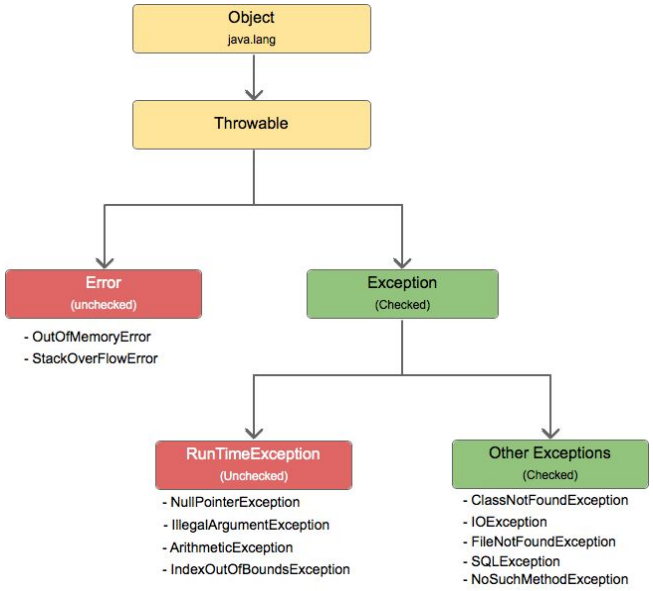
CSE 215L: Programming Language II Lab
Section: 7
Fall 2020

Exception:
Exception handling enables a program to deal with exceptional situations and continue its normal execution.

Types:



Key Words: try, catch, finally, throws, throw

```
main() {
try{
int result = quotient (num1, num2);
System.out.println("result = " + result);
}
catch (ArithmeticException ex){
System.out.println("a number cannot
be divided by 0 ");
}
finally{
System.out.println("this is final clause");
}
```

```
quotient(int num1, int num2) {
If (num2 == 0)
   throw new
ArithmeticException("divisor +cannot be
0");

return num1/num2;
}
```

throw new ArithmeticException(“”) is equivalent to:

```
ArithmeticException ex = new ArithmeticException("");
throw ex;
```

try-catch-finally	Throwable (super class of all exceptions)

try {
// code which may throw any exception}
catch (ExceptionType e){
// handles the exception e}
finally{
// it'll always run}
	Methods to get information about exception:
	
	+getMessage(): String
	+toString(): String
	+printStackTrace(): void
	+getStackTrace(): StackTraceElement[]

CircleWithException.java	CircleWithExceptionMain.java

package exception;
 	
public class CircleWithException {
  private double radius;
 	
  public CircleWithException() {}
  public CircleWithException(double r) {
  setRadius(r);
  }
  public void setRadius(double r) throws IllegalArgumentException{
  if(r >= 0)
  {this.radius = r;}
  else
  {throw new IllegalArgumentException("Radius cannot be negative");
  }
  public double getRadius() {
  return this.getRadius();
  }
  public double getArea() {
  return 3.14*this.radius*this.radius;
  }
}
	package exception;
	
	public class CircleWithExceptionMain {
	
	  public static void main(String[] args) {
	try {
	  CircleWithException circle1 = new CircleWithException(5);
	// System.out.println(circle1.getArea());
	  CircleWithException circle2 = new CircleWithException(-5);
	
	System.out.println(circle1.getArea());
	System.out.println(circle2.getArea());
	}
	catch (IllegalArgumentException ex){
	  System.err.println("negative radius");
	// System.out.println(ex);
	// System.out.println(ex.getMessage());
	// System.out.println(ex.getStackTrace());
	  }
	  }
	}

Write your Exception Class:

RadiusException.java

```
package exception;

public class RadiusException extends Exception{
   private String yourMsg;

   public RadiusException() {
         super();
   }
   public RadiusException(String msg) {
         this.yourMsg = msg;
   }
   public String toString() {
         return this.yourMsg;
   }
}
```

CircleWithException.java

```
package exception;

public class CircleWithException {
   private double radius;

   public CircleWithException() {}
   public CircleWithException(double r) throws RadiusException{
         setRadius(r);
   }
// public void setRadius(double r) throws IllegalArgumentException{
   public void setRadius(double r) throws RadiusException{
         if(r >= 0)
               this.radius = r;
         else
               throw new RadiusException("Radius cannot be negative");
   }
   public double getRadius() {
         return this.getRadius();
   }
   public double getArea() {
```

```
        return 3.14*this.radius*this.radius;
    }
}
```

CircleWithExceptionMain.java

```
package exception;

public class CircleWithExceptionMain {

    public static void main(String[] args) {
        try {
            CircleWithException circle1 = new CircleWithException(5);
//          System.out.println(circle1.getArea());
            CircleWithException circle2 = new CircleWithException(-5);
            System.out.println(circle1.getArea());
            System.out.println(circle2.getArea());
        }
        catch (RadiusException ex){
            System.err.println("negative radius");
            System.out.println(ex);
//          System.out.println(ex.getMessage());
//          System.out.println(ex.getStackTrace());
//          System.out.println(ex.toString());
        }
    }

}
```

Home Task:

Create a Triangle class that extends *Shape*. If the sum of any two sides is not greater than the third side, the Triangle class should throw IllegalArgumentException.