CSE 215: Programming Language II Lab

Lab – 11

Exception Handling

Objective:

- To understand exception and its usage
- To utilize custom exception class

EXCEPTION HANDLING

Exceptions can be handled to prevent the program to be terminated automatically.

Keywords: try, catch, finally, throw, throws

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[]

throw and throws

try{
  if(condition){
  throw new ExceptionType();
  }
}
catch(ExceptionType e){
//handle here
}
	throws is used in method signature
	
	void M() throws ExceptionType{
	  if(condition){
	  throw new ExceptionType();
	  }
	}
	/* The thrown exception must be handled inside caller method */

```
void M() throws Exception1, Exception2, … ExceptionN{
  if(condition){
    throw new Exception1();
  }// else if statements
  else{
    throw new ExceptionN();
  }
}
```

Examples: Try the following codes as practice:

ArrayIndexOutOfBoundsException using **try-catch**	InputMismatchException using **try-catch-finally**

`try{`
`  int[ ] array = new int[5];`
`  array[7] = 20;`
`}`
`catch(ArrayIndexOutOfBoundsException e){`
`  System.err.println("Out of range");`
`}`
	`try{`
	`  int num = scannerName.nextInt();`
	`  System.out.println(num+" is an integer");`
	`}`
	`catch(InputMismatchException e){`
	`  System.out.println("Not an integer");`
	`}`
	`finally{`
	`  scannerName.close();`
	`}`

ArithmeticException using **throw**	NullPointerException using **throws**

`double a = input.nextDouble();`
`double b = input.nextDouble();`
`try{`
`  if(b==0){`
`  throw new ArithmeticException();`
`  }`
`  System.out.println(a/b);`
`}`
`catch(ArithmeticException e){`
`  System.out.println("Invalid value of"`
`+`
`  "b");`
`}`
	`public static void printLength(String s)`
	`throws NullPointerException{`
	`  if(s==null){`
	`  throw new NullPointerException("Null"`
	`+`
	`  "string");`
	`  }`
	`  System.out.println(s.length());`
	`}`
	`public static void main(String[] args) {`
	`  try{`
	`  String str = null;`
	`  printLength(str);`
	`  }`
	`  catch(NullPointerException e){`
	`  System.out.println(e.getMessage());`
	`  }`
	`}`

Creating custom class for Exception

```
public class MyException extends
Exception{
  public MyException(){
    super();
  }
  public MyException(String message){
    super(message);
  }
}
```

```
public void setRadius(double radius)
throws MyException{
  if(radius<0){
    throw new MyException("Invalid"+
    "radius");
  }
  this.radius = radius;
}

public static void main(String[] args)
throws MyException{
  Circle c = new Circle();
  try{
    c.setRadius(-5);
  }
  catch(MyException e){
    System.out.println(e.getMessage());
  }
}
```

Lab Task

1. Design a custom exception class **`InvalidNameException`**.

2. Create a class named **`Patient`** with private data fields name, age and disease.
   a. In **`setName(String name)`** method, throw **`InvalidNameException`** if the name contains less than 3 letters.
   b. In **`setAge(int age)`** method, throw **`IllegalArgumentException`** if the age is negative.
   c. Call the set methods in constructor

3. Create 2 objects with invalid name and age, also one object with valid name and age. Catch the exceptions here.