Object Oriented Programming with Java 18IS34

Unit-4

Packages

- Package is to group related pieces of a program together.
- Serves two purposes
  - related pieces of program can be organised as a unit.
  - Participates in Java's access control mechanism.

Defining a package

- package mypack;
- A hierarchy of packages can be created

```
package bookpack;
class Book {
	private String title;
	private String author;
	private int pubYear;
	Book(String t, String a, int d) {
		title = t;
		author = a;
		pubYear = d;
	}
	void show() {
		System.out.println(title);
		System.out.println(author);
		System.out.println(pubYear);
	}
}
```

```
class BookDemo {
        public static void main(String [] args) {
        Book [] books = new Book[5];
        books[0] = new Book (“CP”, “Knuth”, 1973);
        books[1] = new Book (“SS”, “Mel”, 1851);
        books[2] = new Book (“OS”, “Red”, 1975);
        books[3] = new Book (“ST”, “CI”, 1978);
        books[4] = new Book (“OR”, “Kr”, 1955);
for(int i=0; i<books.length;i++) {
        books[i].show();
        System.out.println();
}
}
}
```

Package and member Access

- members declared public are visible everywhere.
- private is accessible only to the other members of its classs.
- protected is accessible within its package and to all subclasses.

```
package bookpack;
class Book {
	private String title;
	private String author;
	private int pubdate;
	Book(String t, String a, int d) {
		title = t;
		author = a;
		pubDate = d;
	}
	void show() {
		System.out.println(title);
		System.out.println(author);
		System.out.println(pubDate);
	}
}
```

```
package mypack;
class useBook {
        public static void main(String [] args) {
        bookpack.Book [] books = new Book[5];
        books[0] = new Book (“CP”, “Knuth”, 1973);
        books[1] = new Book (“SS”, “Mel”, 1851);
        books[2] = new Book (“OS”, “Red”, 1975);
        books[3] = new Book (“ST”, “CI”, 1978);
        books[4] = new Book (“OR”, “Kr”, 1955);
for(int i=0; i<book.length;i++) {
        books[i].show();
        System.out.println();
}
}
}
```

Understanding protected members

- protected modifier creates a member that is accessible within its package and to subclasses in other packages

```
package bookpack;
class Book {
        protected String title;
        protected String author;
        protected int pubdate;
        Book(String t, String a, int d) {
                title = t;
                author = a;
                pubDate = d;
        }
        void show() {
                System.out.println(title);
                System.out.println(author);
                System.out.println(pubDate);
        }
}
```

- creates extbook of Book

```
package bookpacket;
class Extbook extends bookpack.Book {
        private String condition;
        public Extbook(String t, String a, int d, String c) {
                super(t,a,d);
                condition = c;
        }
        public void show() {
        super.show();
        System.out.println("Condition" + condition);
        public String getcondition() { return condition; }
        public void setcondition(String c) { condition = c; }
        gettitle()....settitle().....getauthor()....setauthor()...getdate()...setdate()...
        }
}
```

```
class protectedDemo{
    public static void main(String [] args) {
    Extbook [] books = new Extbook[5];
    books[0] = new Extbook(“A”, “B”, 1973, “were used”, “moby”);
    .
    .
    .
    for(int i=0; i<books.length ; i++)
        if(books[i].gettitle() ==”moby”)
            System.out.println(book[i].getcondition());
    }
}
```

importing packages

- import pkg.classname;
- Ex: package mypack;

```
import bookpack.*;
// use Book class from bookpack
```

importing Java's standard packages

Subpackage	Description
java.lang	contain a large no of general purpose classes
java.io	contain I/O classes
java.net	contain classes for networking
java.awt	contain classes for abstract window toolkit
java.util	contain classes for collections framework

static import

- It imports static members of a class or interface.

```
import static java.lang.Math.sqrt;
import static java.lang.Math.pow;
class quadratic{
    public static void main(String [] args){
        double a,b,c,x;
        a=4; b=1; c=-3;
        x = (-b + sqrt(pow(b,2) -4*a*c))/(2*a);
        System.out.println("First Soltn" + x);
        x = (-b - sqrt(pow(b,2) -4*a*c))/(2*a);
        System.out.println("Second Soltn" + x);
    }
}
```

Exception Handling

- An exception is an error that occurs at runtime.
- Exception handling: streamlines error handling
- Exception handler: executed automatically when an error occurs.
- defines standard exceptions for common program errors.
- All exceptions are derived from a class called throwable
- Exception handling is managed by five keywords try,catch,throw,throws and finally.
- To manually throw an exception use keyword throw.

General form

```
try
{
        //block of code
}
catch(Exception1 exob)
{
//handler for except1
}
catch(Exception2 exob)
{
//handler fro excep2
}
.
.
.
```

Simple Example

```
class ExcDemo1 {
        public static void main(String [] args) {
                int [] nums = new int [4];
                try{
                System.out.println(“Before Execution is genetrated”);
                nums[7] = 10;
                System.out.println(“THis wont be displayed”);
        }
        catch(ArrayIndexOutOfBoundException exe) {
        System.out.println(“Index out of bound “);
        }
        System.out.println(“After catch”);
}
}
```

Consequences of uncaught exceptions

```
class ExcDemo{
    public static void main(String [] args) {
        int [] numer = {4,8,16,32,64,128,256};
        int [] denom = {2,0,4,4,0,8};
        for(int i=0;i< numer.length;i++)
        try
        {
            System.out.println(numer[i] + “/” + denom[i] +”is” + numer[i]/denom[i]);
        }
        catch(ArithmeticException exc) {
            System.out.println(“Cant divide by zero”);
        }
        catch(ArrayIndexOutOfBoundException exc) {
            System.out.println(“No matching element found”);
        }
    }
}
```

catching subclass exceptions

```
class ExcDemo{
    public static void main(String [] args) {
        int [] numer = {4,8,16,32,64,128,256,512};
        int [] denom = {2,0,4,4,0,8};
        for(int i=0;i< numer.length;i++)
        try
        {
            System.out.println(numer[i] + "/" + denom[i] +"is" + numer[i]/denom[i]);
        }
        catch(ArrayIndexOutOfBoundException exc) {
            System.out.println("No matching element found");
        }
        catch(Exception exc) {
            System.out.println("some exception occured");
        }
    }
}
```

try blocks can be nested

```
try {
      for(int i=0;i<numer.length;i++) {
      try {
            System.out.println(numer[i]/denom[i]);
      }
      catch(ArithmeticException exc) {
            System.out.println("Cant divide by zero");
            }
        }
  }
     catch(ArrayIndexOutOfBoundException exc)
      {
            System.out.println("No matching element found");
      }
```

Throwing an exception

```
class ThrwDemo{
    public static void main(String [] args) {
        try
        {
            System.out.println(“Before throw”);
            throw new ArithmeticException();
        }
        catch(ArithmeticException exc) {
            System.out.println(“Exception caught”);
        }
        System.out.println(“After try/catch block”);
        }
}
```

Rethrowing an Exception

```
class Rethrow{
        public static void genException() {
                int [] numer = {4,8,16,32,64,128,256,512};
                int [] denom = {2,0,4,4,0,8};
                for(int i=0;i< numer.length;i++)
                try
                {
                        System.out.println(numer[i]/denom[i]);
                }
                catch(ArithmeticException exc) {
                        System.out.println(“Cant divide by zero”);
                }
                catch(ArrayIndexOutOfBoundException ex() {
                        System.out.println(“No matching element found”);
                        throw exc;    //rethrow exception
                }
        }
}
```

```
class RetrowDemo {
    public static void main(String [] args) {
        try{
        Rethrow.genException();
        }
        catch(ArrayIndexOutOfBoundException exc) {
            System.out.println(“Fatal Error” + “Program terminated”):
    }
    }
}
```

Closer look at throwable

Throwable fillinStackTrace(): returns a throwable object that contains a completed stacktrace.

String getLocalisedMessage() : returns localised description of exception.

void PrintstackTrace() : displays the stack trace

void printStackTrace(PrintStream stream): sends the stack trace to the specified stream.

void printstackTrace(PrintWriter stream): sends the stacktrace to the specified stream.

String toString(): return a string object containing complete description of exception.

Using finally

- finally block is executed whenever execution leaves a try block no matter what condition causes it.

```
class usefinally{
        public static void genException(int what){
                int t;
                int [] nums = new int[2];
                System.out.println("Receiving" + what);
                try{  switch(what) {
                        case 0: t=10; t=t/what;  break;
                        case 1: nums[4]=4; break;
                        case 2: return;
                        }
                }
                catch(ArithmeticException exc){
                        System.out.println("cant divide by zero");       return;
                        }
                catch(ArrayndexOutofBoundException exc){    System.out.println("No matching element found"):
                }
                finally{
                        System.out.printlln("leaving try");
```

```
class finallyDemo{
    public static void main(String [] args){
        for(int i=0;i<3;i++){
            usefinally.genException(i);
            System.out.println();
        }
    }
}
```

Using throws and builtin exceptions

- If a method generates an exception that it does not handle it must declare that exception in a throws clause.

```
ref_type Methname(Param_list) throws exceplist{
        //body
}
```

```
class throwsDemo {
        public static char prompt(String str)
                throws java.io.IoException {
                System.out.print(str + “:”);
                return(char) System.in.read();
                }
public static void main(String [] args){
        char ch;
        try{
                ch=prompt(“enter a letter”);
        }
        catch(java.io.IoException exc) {
                System.out.println(“I/O Exception generated”);
                ch = 'X';
        }
        System.out.println(“You pressed” +ch);
        }
}
```

- Java built in Exceptions:

  Arithmetic,ArrayIndexOutOfBoundException, ArraysforException
- Creating Exception Subclasses:
  - Builtin Exception class can be extended