**Lecture:**

**[ Review Chapter 11: Inheritance & Interfaces ]**

**An object's interface with the outside world is formed by methods; the buttons on the face of your cellular smartphone, for example, are the interface between you and the electrical circuits beneath the class on the phone.**

[**http://docs.oracle.com/javase/tutorial/java/concepts/interface.html**](http://docs.oracle.com/javase/tutorial/java/concepts/interface.html)

**interface Bicycle {**

**// wheel revolutions per minute**

**void changeCadence(int newValue);**

**void changeGear(int newValue);**

**void speedUp(int increment);**

**void applyBrakes(int decrement);**

**}**

/\* File name : MammalInt.java \*/

**public** **class** MammalInt **implements** Animal{

**public** **void** eat(){

System.*out*.println("Mammal eats");

}

**public** **void** travel(){

System.*out*.println("Mammal travels");

}

**public** **int** noOfLegs(){

**return** 0;

}

**public** **static** **void** main(String args[]){

MammalInt m = **new** MammalInt();

m.eat();

m.travel();

}

}

//Animal.java

**public** **interface** Animal {

**public** **void** eat();

**public** **void** travel();

}

**[ Review Chapter 17: Generics ]**

**• Introduction to Generics**

**• Writing a Generic Class**

**• Why Use Generics?**

[**http://docs.oracle.com/javase/tutorial/java/generics/why.html**](http://docs.oracle.com/javase/tutorial/java/generics/why.html)

**[ Example : Generic Stack Class ]**

//Stack.java

**import** java.util.Arrays;

**public** **class** Stack <T>

{

**private** **int** count;

**private** T[] data;

**public** Stack()

{

data = (T[]) **new** Object[8];

count = 0;

}

**void** expandCapacity()

{

data = Arrays.*copyOf*(data, data.length \* 2);

}

**void** push(T e)

{

**if** (count == data.length)

expandCapacity();

data[count++] = e;

}

T pop() **throws** Exception

{

**if** (count <= 0)

{

**throw** **new** Exception("stack empty");

}

count--;

**return** data[count];

}

**boolean** isEmpty()

{

**return** count == 0;

}

**int** size()

{

**return** count;

}

**public** **static** **void** main(String[] args) **throws** Exception

{

Stack<String> s = **new** Stack<String>();

s.push("Alice");

s.push("Bob");

s.push("Carl");

s.push("Dave");

**while** (!s.isEmpty())

System.*out*.println(s.pop());

}

}

**[ Chapter 18: Collections ]**

**-Introduction to the Java collections Framework**

**-Lists**

**-Sets**

**-Maps**

**-The Collections Class**

**Set: a collection with no notion of position within the collection for stored elements. Sets do not permit duplicate elements.**

**[ Set Operations - On One Set ]**

**Operations on a particular set include:**

**• Membership: is an item in the set?**

**• Insert: adding an item to the set**

**• Remove: take an item from the set**

**• Size: the number of items in the set**

**[ Arrays and the ArrayList Class ]**

**The ArrayList Class**

**Similar to an array, an ArrayList allows object storage**

**Unlike an array, an ArrayList object:**

**-Automatically expands when a new item is added**

**-Automatically shrinks when items are removed**

**Requires:   
  
import java.util.ArrayList;**

**Lab:**

**[ Review Lab 4: Stock Market Application ]**

**• Observe the grading details and rubric on the lab assignment**

**[ Review Lab 5: Sequential File Processing ]**

**• Sequential versus Random**

**• Input for Reading**

**• Output for Writing**

**• Append Mode ( adding content to the end of the file )**

|  |  |  |
| --- | --- | --- |
| **employee name** | **hours** | **wage** |
| David Davies | 30 | 8.75 |
| Eddie Edwards | 40 | 7.75 |
| Fran Francis | 35 | 11.20 |
| Gina Georges | 22 | 6.30 |
| Hyacinth Ho | 40 | 9.21 |

**Here are the steps for simple file processing:**

**Writing to a File**

**declare a file name**

**declare a file processing object**

**store some text or numbers in a local variable**

**write the contents of the variable in the text file**

**close the file processing object**

**Reading From a File**

**declare a file name**

**declare a file processing object**

**read some text or numbers in a local variable**

**display the contents of the variable on the screen**

**close the file processing object**