JAVA Basic Concept

TUTORIAL 1

DBMS 2013-2014 Fall

TAs:Nagehan İlhan

Mahiye Uluyağmur

JAVA Classes

- “class” keyword is used to define JAVA classes
- Classes have “interfaces” to the outside world
- All classes are defined in `.java` files
  - Every file can only have one top level public class
- Classes have two types of members
  - Attributes
  - Methods (constructor, destructor, setter/getter)
- To create an object from a class “new” keyword is used

JAVA Classes

```
package com.graphLib;
public class Node {
    private String name;
    public Node(String name) {
        this.name=name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name=name;
    }
}
```

JAVA Classes

JAVA	C++

import com.graphLib.Node;
 	
public class Graph{
 	
private String name;
private Node[] nodes;
private int nOfNodes;
 	
public static int nOfGraphs;
public Graph(){.......}
public Graph(int nOfNodes){...}
public Graph(final Graph aGraph) {.......}
public boolean addNode(String nodeName) {.....}
public boolean deleteNode(String nodeName) {.....}
private int containsNode(String name){..... }
}
	class Graph {
	
	private:
	string name;
	Node* nodes;
	int nofNodes;
	int containsNode(string);
	public:
	static int nOfGraphs;
	
	Graph();
	Graph(int);
	Graph(const Graph &);
	~Graph();
	bool addNode(string);
	bool deleteNode(string);
	};

JAVA Classes

- JAVA is a “pure object oriented“ programming language (platform)
  - Every class is inherited from another class except the object class
- Object class has two important methods
  - Clone()
  - toString()
- “garbage collector” is used to manage memory in the object lifecycle

Data handling I

- “public” and “private” keywords are used while defining data
- Setter/getter methods are used to private variables

```
private T var;
public void setVar(T var){
            this.var=var;}
public T getVar(){
return var;}
```

Label	Class	World
public	Y	Y
unlabeled	Y	N
private	Y	N

Packages

- bundle groups of related types into packages
  - organize a bunch of classes and interfaces into a *package*
  - defines a namespace that contains all the classes
- different packages may have classes that have the same name
  - Two different packages may have classes named as “print” that have different capabilities, packages enable programmers to use these classes where necessary
- “import” keyword is used to import packages

Data handling II

```
package mat1;

public class AnaSinif {
   public int Toplam(int ilksayi, int ikincisayi) {
       return ilksayi+ikincisayi;
   }
}
public class Uslu {
    public double ussu (double sayi, double usDegeri) {
         double toplam=1;
         for(int i=1;i<=usDegeri;++i)
             toplam=sayi*toplam;
         return toplam;
    }
}
```

```
package mat2;

public class AnaSinif {
    public void KacKarakter(String metin) {
        System.out.println(metin.length());
    }
}
```

```
import mat1.*;
import mat2.*;

public class PaketKavrami{

  public static void main(String[] args) {
      Uslu u1=new Uslu();
      double sonuc=u1.ussu(3,2);
      System.out.println(sonuc);

      mat1.AnaSinif t1=new mat1.AnaSinif();
      int toplam=t1.Toplam(24,12);
      System.out.println(toplam);
     mat2.AnaSinif   t2=new mat2.AnaSinif();
      t2.KacKarakter("yazdigim bu cümlenin karakter sayısı
ne?");
  }
}
```

Label	Class	World	Package
public	Y	Y	Y
unlabeled	Y	N	Y
private	Y	N	N

Static members

- To use a classes attributes and methods without creating an object, “static” keyword must be used for such attributes and methods
- *static* method usage:
  - A method that will not be called from an object but perform a task can be written as static methods

```
public class StatikMethod{
    public static void calis(){
        ……………………………
    }
}
StatikMethod.calis();
```

- *static* attribute usage:
  - there is exactly one copy of such variables in existence
  - Variable value can change during the coarse of a run, but only the last value is stored
- Constants are defined with “final” keyword

```
static final double PI = 3.141592653589793;
```

Static members

```
public class Node{

    private static int nOfNodes=0;
    private String name;

    Public Node(){
        Node.nOfNodes++;
        this.name=”Node”+Node.nOfNodes;
     }

     Public Node(String name){
         Node.nOfNodes++;
         this.name=name;
     }
}
```

Inheritence

- Object-oriented programming allows classes to *inherit* commonly used state and behavior from other classes
- “extends” keyword is used

```
public class Dikdortgen extends Dortgen
```

- Java doesn't provide multiple inheritance unlike C++
  - It is possible to use nested derivation

Data handling III

public class Dikdortgen extends Dortgen

- Class of Dikdortgen has same properties with class of Dortgen .

But the reverse is not true.

- Restrictions on access controllers are used in inheritance of class variables and methods as used in class definitions.
- Super class methods and variables are called from sub classes by using the key word «super».

Label	Class	World	Package	Sub Class
public	Y	Y	Y	Y
unlabeled	Y	N	Y	N
private	Y	N	N	N
protected	Y	N	Y	Y

Data handling III

```
class GeometrikSekil{

    double Taban;
    double Yukseklik;
    public GeometrikSekil(double a,
    double b)
    {
        Taban=a;
        Yukseklik=b;
    }
}
```

```
class Ucgen extends GeometrikSekil {

    String UcgenTuru;
    public Ucgen(double tb,double yuk,
    String tip)
        {
        super(tb,yuk);
        UcgenTuru=tip;
        }
    public double Alan(){
        return (Taban*Yukseklik)/2;
    }
}
```

```
public class KalitimOrnegi{
    public static void main(String[] args){
        Ucgen u=new Ucgen(10,8.66,"Eskenar");
        System.out.println("Ucgen alani "+u.Alan());
    }
}
```

Software Development

- Java (except for primitive types) is completely object-oriented programming language.
- Java softwares consist of a set of class definitions.
- Executable Java Classes have a method which is
  - public static void main (String args[])
- «init» operations are performed in only «main» method.

Software Development

```
package PointLib;
public class Point {
	private String name;
	private int corX;
	private int corY;
	public Point(String name){
	this.name=name;
	corX=0;
	corY=0;
}
public Point(String name,int x,int y)
{
this.name=name;
this.corX=x;
this.corY=y;
}
```

```
public void moveUp(){
		corY++;
}
public void moveDown(){
		corY--;
}
public void moveLeft(){
		corX--;
}
public void moveRight(){
		corX++;
}
public String toString(){
		return
name+":("+corX+","+corY+")"; }}
```

Software Development

```
package PointLib;
public class Deneme {
public static void main (String args[]) {
    Point aPoint=new Point("A");
    aPoint.moveUp();
    aPoint.moveRight();
    System.out.println(aPoint);
    aPoint.moveDown();
    aPoint.moveLeft();
    System.out.println(aPoint);
}
}
```

Variable Access

Primitive types are defined with values, other types are defined with references.

Parameters are being send always with values.

Attention:

- When a non primitive type is being send to a method as a parameter, the reference of type is being send as a value.

Clone() method is too important on changing values among objects.

Variable Access

```
public class Deneme {
    public static void main (String args[]) {
        int x=3;
        int y=x;
        x=5;
        System.out.println(“y=” + y);
    }
}
y=3;
```

Variable Access

```
public class Deneme {
    public static void main (String args[]) {
        Point aPoint=new Point("A");
        Point bPoint=aPoint;
        aPoint.moveUp();
        aPoint.moveRight();
        System.out.println(bPoint);
    }
}
```

```
A:(1,1)
```

Variable Access

```
public class Deneme {
    public static void main (String args[]) {
        Point aPoint=new Point("A");
        movePoint(aPoint);
        System.out.println(aPoint);
    }
    private static void movePoint(Point aPoint){
        aPoint.moveUp();
        aPoint.moveRight();
        Point bPoint=new Point("B");
        bPoint.moveDown();
        bPoint.moveLeft();
        System.out.println(bPoint);
        aPoint=bPoint;
    }
}
```

Collections

- Array sizes can vary for multi-dimensional arrays in JAVA.

Collections

```
package PointLib;
import java.util.Arrays;
public class Deneme {
    public static void main (String args[]) {
        int[][] birDizi={
            {1,2,3},
            {4},
            {},
            {5,6}
    };
        for(int i=0;i<birDizi.length;i++)
         for(int j=0;j<birDizi[i].length;j++)
                System.out.print(birDizi[i][j]+" ");
        System.out.println();
//Output: 1 2 3 4 5 6
```

Collections

```
// OR
for(int[] altDizi:birDizi)
        for(int i: altDizi)
                System.out.print(i+" ");
System.out.println();
// OR
for(int[] altDizi:birDizi)
        System.out.print(Arrays.toString(altDizi)+" ");
System.out.println();
        System.out.println(Arrays.toString(birDizi));
}
}
//Output: 1 2 3 4 5 6
// Output : [1, 2, 3] [4] [] [5, 6]
// Output : [[I@1fee6fc, [I@1eed786, [I@187aeca, [I@e48e1b]
```

Collections

- It is useful to use pre-defined Java collections classes.

Collections

```
package PointLib;
import java.util.ArrayList;
import java.util.LinkedList;
public class Deneme {
public static void main (String args[]) {
        ArrayList<Point> pointList = new ArrayList<Point>();
        pointList.add(new Point("A"));
        pointList.add(new Point("B"));
        pointList.add(new Point("C"));
        for(Point p:pointList)
                p.moveUp();
        for(int i=0;i<pointList.size();i++)
                System.out.println(pointList.get(i));
```

Collections

```
LinkedList<Point> pointLinked = new LinkedList<Point>();
pointLinked.addAll(pointList);
System.out.println(pointLinked.getFirst());
System.out.println(pointLinked.getLast());
}
}
```

For Additional Information

- http://download.oracle.com/javase/tutorial