### Learning Activities – ARRAYS - The Basics

**CORE**

This session covers arrays and common array processing algorithms. The next session will cover arrays of objects.

1. **READ**

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html>

1. **DO: INPUT and OUTPUT an array**.

Create the following array demo class called ArrayDemo.

|  |
| --- |
| import java.util.Scanner;  public class ArrayDemo{  public static void main(String[] args) {    //declare a constant for the MAX elements in the array  final int MAX = 5;    Scanner kbd = new Scanner(System.in);  // options to declare and create the array  // declare variable first  // int marks[];  // then create array  // marks = new int[MAX];  // then set the value (mark in this case) of each element  // // or declare the variable and create the array all in one line // int marks[] = new int[MAX];  // // or you can put the square brackets after the datatype  // int[] marks = new int[MAX];  // // or  // declare variable and define size by defining the data, everything is happening at once here // including the size of the array through the number of data items given (5 in this case)  // int[] marks= {10,11,10,10,12};  //Lets do it this way  int marks[] = new int[MAX];  // You can set the data for each element of the array using the index in the square brackets  // eg marks[0] is the zeroth element of the marks array  marks[0] = 10;  marks[1] = 11;  marks[2] = 10;  marks[3] = 10;  marks[4] = 14;  System.out.println( "All marks have now been entered");  // print out all the marks  // The array now knows how much data it can take through the use of - array.length  System.out.println("\nPrinting all the marks set in the code:");  for(int i=0;i< marks.length; i++) {  System.out.print(marks[i] + " ");  }  //Now lets get the user to enter the marks  //input students marks  for (int i=0; i<marks.length; i++ ){  System.out.println("Enter the students mark :");  marks[i] = kbd.nextInt();  }  System.out.println( "All marks have now been entered");  // print out all the marks  System.out.println("\nPrinting all the marks entered by the user:");  for(int i=0;i< marks.length; i++) {  System.out.print(marks[i] + " ");  }  }  } |

Let’s now do some calculations with the marks entered:

**Calculate AVERAGE**

Now add in this code to main that will calculate the average of the marks as follows:

|  |
| --- |
| int total =0;  float average = 0.0F;  // calculate average  //first calculate total  for(int i=0; i<marks.length; i++ ){  total += marks[i];  }  average = (float)total/marks.length; //need to typecast otherwise int division  System.out.println("\nThe average of all marks is :" + average); |

**Search For MAXIMUM**

Now we want to find the maximum mark in the array. Add the following code to the end of main in ArayDemo:

|  |
| --- |
| int index = 0;  int maxMark = 0;  int maxMarkIndex = 0;  // search through all the marks to find the maximum mark and the student that got the max score  maxMark = marks[0]; // set max score to first mark  maxMarkIndex = 0; //set maxMarkIndex to first student mark  // start search from second element  for (index = 1; index < 5; index++) {  if (marks[index] > maxMark) {  maxMark = marks[index];  maxMarkIndex = index;  } //endif  } //endfor  System.out.println("\nStudent " + maxMarkIndex + " achieved the highest score of " + maxMark); |

1. **Use of for each loop**  
   Create a **Card** class with instance variables String suit and String rank

|  |
| --- |
| public class Card {  public static final String DEFAULT\_SUIT = "none";  public static final String DEFAULT\_FACEVALUE = "none";  private String suit;  private String faceValue;  public Card() {  this(DEFAULT\_SUIT, DEFAULT\_FACEVALUE);  }  public Card(String suit, String rank) {  this.suit = suit;  this.faceValue = rank;  }  public String getSuit() {  return suit;  }  public void setSuit(String suit) {  this.suit = suit;  }  public String getFaceValue() {  return faceValue;  }  public void setFaceValue(String faceValue) {  this.faceValue = faceValue;  }  @Override  public String toString() {  return "[" + faceValue + " of " + suit + ']';  }  } |

Now create a main class called **CardGame** that creates a deck of cards, firstly creating an array of suits and array of ranks, then create the deck:

|  |
| --- |
| public class CardGame {  public static void main(String[] args) {  String[] suits = new String[4];  // set the suits  suits[0] = "Clubs";  suits[1] = "Diamonds";  suits[2] = "Hearts";  suits[3] = "Spades";  // set the face value  String[] faceValues = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};  // create a deck of cards as an array of card objects  Card[] deck = new Card[52];    /\*int index = 0;  for (int suit = 0; suit <= 3; suit++) {  for (int faceValue = 0; faceValue < 13; faceValue++) {  deck[index] = new Card(suits[suit], faceValues[faceValue]);  System.out.println(deck[index]);  index++;  }  }  \*/  // better to use a for each loop  int index = 0;  for (String suit : suits) {  for (String faceValue : faceValues) {  deck[index] = new Card(suit, faceValue);  index++;  }  }  } |

Add the output of each card. This could have been done using a for loop:

for (int i=0; i<deck.length; i++)

{

System.out.println(deck[i]);

}

But can be written using a for each loop

|  |
| --- |
| for (Card card : deck) {  System.out.println(card);  } |

Now you could generate a random card

|  |
| --- |
| // choose a random card from the deck  int r = (int) Math.round(Math.random() \* 51);  System.out.println("\nRandom card " + deck[r]); |

1. **Working with an array of Objects  
     
   Create an array of Dog objects**.

Using the previous **Dog** class, create a **DogArrayTest** class as follows:

|  |
| --- |
| public static void main(String[] args) {  Dog[] myDogs = new Dog[3]; //declare an array of 3 dog references  myDogs[0] = new Dog();  myDogs[1] = new Dog("D001", 55, "Lab", "Butch");  myDogs[2] = myDogs[0]; //set reference to myDogs[0]  myDogs[2].setSize(80); //this will set the size for myDogs[0] as well  //loop through array to output each dog and call bark method  for (int i = 0; i < myDogs.length; i++) {  System.out.println(myDogs[i] + "\nbarks " + myDogs[i].bark());  }  // now create a new array of 10 dogs using the no-arg constructor  Dog[] dogArray = new Dog[10];  for (int i = 0; i < dogArray.length; i++) {  dogArray[i] = new Dog();  }  // output the array using a for loop  for (int i = 0; i < myDogs.length; i++) {  System.out.println(myDogs[i]);  }  // output the array using a for each loop  for (Dog myDog : myDogs) {  System.out.println(myDog);  }  } |

1. Using the previous **Book** class, create a **Bookcase** class to declare an array called myBooks of 10 books. Use a for loop to create each Book using the no-arg constructor. Use a for each loop to output the books.
2. Using the previous **Rectangle** class:  
   2 float instance variables width and length, all arg and no arg constructors, getters, setters, toString(), a method calcArea() and a method calcPerimeter().   
      
   Use a for loop to create an array of 8 rectangles in your **RectangleTest** class with lengths and widths of 1 and 2, 2 and 3, 3 and 4, 4 and 5, 5 and 6, 6 and 7, 7 and 8, 8 and 9. (hint use new Rectangle(i+1,i+2))

Use a for each loop to output each rectangle’s length, width, area and perimeter.

1. **Arrays are not always full**. **The need for a variable to count the number of items in the array.**  
   Arrays are not always filled with data, for example, the length of the array may be 100 to allow that much to be stored but it may only contain 20 items at a given point in the code.  
     
   When this is the case you need a variable to help us know how many items there actually is and you use that variable in the for loops rather than the length of the array. The following example will demonstrate this using an array of Unit objects and the counter numberOfUnits.  
     
   Using your previously created **Unit** class, create a new class called **BlockOfUnits** that will implement an array of unitsand **BlockOfUnitsTest** with code as follows:

|  |
| --- |
| /\*\*  \* Implements a collections of Units (Apartments)  \* This implementation uses Aggregation. Units are always referred to via the  \* unit number (eg 1A).  \*  \*/  public class BlockOfUnits {  public static final int DEF\_NUM\_UNITS = 5;  private int numberOfUnits; // The number of defined units in the block.  // This goes up by one every time a unit is added.  private Unit unit[]; // The array of units  /\*----------------------------------------------------------------------\*/  /\*\*  \* Create a block of units of the DEFAULT size BUT containing NO units. Units  \* will need to be added to the block using the addUnit method  \*  \*/  public BlockOfUnits() {  this(DEF\_NUM\_UNITS);  }  /\*\*  \* Create a block of units of the provided size BUT containing NO units. Units  \* will need to be added to the block using the addUnit method  \*  \* @param maxNumberOfUnits - the number of units that can be added to the block  \*/  public BlockOfUnits(int maxNumberOfUnits) {  this.numberOfUnits = 0;  this.unit = new Unit[maxNumberOfUnits];  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* @return Returns the number of defined units in the block  \*/  public int getNumberOfUnits() {  return this.numberOfUnits;  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* Adds one unit to the block. Will not add the unit if there is no room left  \* in the block  \*  \* @param u - the unit to be added  \* @return none  \*/  public void addUnit(Unit u) {  if (this.numberOfUnits < this.unit.length) {  this.unit[this.numberOfUnits] = u;  this.numberOfUnits++;  }  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* Records a unit as rented for a certain amount of weekly rent  \*  \* @param unitNumber - the unit number of the unit  \* @param rentAmount - the amount of the weekly rent  \*/  public void markAsRented(String unitNumber, double rentAmount) {  int index;  index = findUnit(unitNumber);  if (index != -1) {  this.unit[index].setRented(true);  this.unit[index].setRentAmount(rentAmount);  }  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* Records a unit as NOT rented ie vacant  \*  \* @param unitNumber - the unit number of the unit  \*/  public void markAsVacant(String unitNumber) {  int index;  index = findUnit(unitNumber);  if (index != -1) {  this.unit[index].setRented(false);  }  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* Finds the unit with a particular unit number in the block of units  \*  \* @param unitNum - the unit number of the unit  \* @return the index in the array of units of the unit with unit number unitNum  \* will return -1 if there is no unit with number unitNum.  \*/  private int findUnit(String unitNum) {  int foundAt;  Loop while within arraybounds and unitNumber not found  int index;  foundAt = 0;  while (foundAt < this.numberOfUnits  && !(this.unit[foundAt].getUnitNumber().equals(unitNum))) {  foundAt++;  }  if (foundAt < this.numberOfUnits) {  index = foundAt;  } else {  index = -1;  }  return index;  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* Find all vacant units in the block.  \*  \* @return an array if the empty units. The array will be empty if there are  \* no empty units.  \*/  public Unit[] findVacantUnits() {  Unit[] freeUnits;  int count;  int i;  freeUnits = new Unit[this.numFreeUnits()];  count = 0;  for (i = 0; i < this.numberOfUnits; i++) {  if (!this.unit[i].isRented()) {  freeUnits[count] = this.unit[i];  count++;  }  }  return freeUnits;  }  /\*----------------------------------------------------------------------\*/  /\*\*  \* Find the number of vacant units in the block.  \*  \* @return the count of empty (not rented)  \*/  public int numFreeUnits() {  int i;  int numFree;  numFree = 0;  for (i = 0; i < this.numberOfUnits; i++) {  if (!this.unit[i].isRented()) {  numFree++;  }  }  return numFree;  }  /\*----------------------------------------------------------------------\*/  public String toString() {  String temp;  temp = "[Block size=" + this.unit.length + ", Number Of Units=" + numberOfUnits;  for (int i = 0; i < this.numberOfUnits; i++) {  temp = temp  + ",unit[" + i + "]="  + "," + this.unit[i];  }  temp = temp + "]";  return temp;  }  } |

**To test the BlockOfUnits create the following BlockOfUnitsTest class**

|  |
| --- |
| public class BlockOfUnitsTest {  /\*----------------------------------------------------------------------\*/  public static void main(String args[]) {  BlockOfUnits bou;  Unit u1 = new Unit("1", 10, 100.00, 10.0);  Unit u2 = new Unit("45A", 4, 123.45, 23.0);  // Test no-arg constructor  System.out.println("\nTest no-arg constructor");  bou = new BlockOfUnits();  System.out.println(bou);  // Test all-arg constructor - block size 10  System.out.println("\nTest all-arg constructor - block size 10");  bou = new BlockOfUnits(10);  System.out.println(bou);  // Test adding two units  System.out.println("\nTest adding two units");  System.out.println("Unit 1 = " + u1);  System.out.println("Unit 2 = " + u2);  bou.addUnit(u1);  bou.addUnit(u2);  System.out.println(bou);  // Test adding more than the number of units allowed in the block - try adding 3 units to a block of size 2  System.out.println("\nTest adding more than the number of units allowed in the block - try adding 3 units to a block of size 2");  bou = new BlockOfUnits(2);  bou.addUnit(u1);  bou.addUnit(u2);  bou.addUnit(new Unit("2", 99, 900.00, 90.0));  System.out.println(bou);  // Test setting unit with number 45A to being rented at $200 per week  System.out.println("\nTest setting unit with number 45A to being rented at $200 per week");  bou.markAsRented("45A", 200);  System.out.println(bou);  // Test setting unit with number 45A to being vacant  System.out.println("\nTest setting unit with number 45A to being vacant");  bou.markAsVacant("45A");  System.out.println(bou);  // Test counting and finding vacant units. Should find 45A and 2 as vacant.  System.out.println("\nTest counting and finding vacant units. Should find 45A and 2 as vacant.");  bou = new BlockOfUnits(10);  bou.addUnit(u1);  bou.addUnit(u2);  bou.addUnit(new Unit("2", 99, 900.00, 90.0));  bou.markAsRented("1", 10);  bou.markAsVacant("2");  bou.markAsVacant("45A");  Unit[] free;  System.out.println("Number of free units = " + bou.numFreeUnits());  free = bou.findVacantUnits();  for (int i = 0; i < free.length; i++) {  System.out.println("Unit vacant -> " + free[i]);  }  }  } |

Add 2 more units to the BlockOfUnitsTest class.

1. Update the Student class so that a student can have an array of subjects:

|  |
| --- |
| public static final String DEFAULT\_STUDENTID = "DEF ID";  public static final int DEF\_NUM\_SUBJECTS = 5;  private String studentID;  private int numSubjects;  // student "has-a" array fo subjects  private Subject[] subjects;  //all-arg constructor  public Student(String studentID, int maxSubjects) {  this.studentID = studentID;  this.numSubjects=0;  this.subjects = new Subject[maxSubjects];  }  //one-arg constructor  public Student(String studentID) {  this(studentID, DEF\_NUM\_SUBJECTS);  }  //no-arg constructor  public Student() {  this(DEFAULT\_STUDENTID, DEF\_NUM\_SUBJECTS);  }  public String getStudentID() {  return studentID;  }  public void setStudentID(String studentID) {  this.studentID = studentID;  }  public int getNumSubjects() {  return numSubjects;  }  public void setNumSubjects(int numSubjects) {  this.numSubjects = numSubjects;  } |

Create a method addSubject(Subject subject) to add subjects for a student:

|  |
| --- |
| //add a subject  public void addSubject(Subject subject) {  if (this.numSubjects < this.subjects.length) {  this.subjects[this.numSubjects] = subject;  this.numSubjects++;  }  } |

Create a toString() that will output the subject array

|  |
| --- |
| public String toString() {  String temp = "";  for (int i = 0; i < this.numSubjects; i++) {  temp = temp  + ",subject[" + i + "]="  + "," + this.subjects[i];  }  temp = "[StudentID " + this.studentID + " has " + this.numSubjects  + " subjects: \n" + temp + "]";  return temp;  } |