**Core JAVA Assignment Questions: Part 1**

**Fundamental Programming Structure in Java**

1. Write a program in Java to check if a number is even or odd in Java? (input 2 output true, input 3 : output false)

2. Write a program in Java to find out if a number is prime in Java? (input 7: output true , input 9 : output false)

3. Write Java program to check if a number is palindrome in Java? (121 is palindrome, 321 is not)

4. How to find if a number is power of 2 in Java? (1, 2, 4 power of 2, 3 is not)

5. Write program to sort an integer array without using API methods?

6. Write Java program to check if a number is Armstrong number or not? (input 153 output true, 123 output false)

7. Write a program in Java to reverse any String without using StringBuffer?

8. Write a program in Java to print Fibonacci series up to given number? Write both iterative and recursive version.

9. Write a java program to input a string and find the total number of uppercase letters in the string?

10. Write a function that simulates rolling a pair of dice until the total on the dice comes up to be a given number. The number that you are rolling for is a parameter to the function. The number of times you have to roll the dice is the return value of the function. You can assume that the parameter is one of the possible totals: 2, 3, ..., 12. Use your function in a program that computes and prints the number of rolls it takes to get snake eyes. (Snake eyes means that the total showing on the dice is 2.)

**Part #1**

1. Create a class called “Assignment02a”

2. Solve the following problem: Write a program that prompts the user to enter two points (x1, y1) and (x2, y2). Calculate and display their distance using the following formula. You can use Math.pow(a, 0.5) to compute the square root of an expression. Math.pow() returns a double.

|  |
| --- |
| Enter x1: 1.5  Enter y1: -3.4  Enter x2: 4  Enter y2: 5  Distance between these points is: 8.764131445842194 |

**Part #2**

1. Create a class called “Assignment02b”

2. Solve the following problem: Write a program that asks the user to enter in their name as well as a length value that represents the length of their dorm room, in feet. Then convert this value into meters, yards, inches and centimeters and display the result to the user as follows:

|  |
| --- |
| What's your name? Craig  How long is your dorm room, in feet? 10  In meters, that's \_\_\_\_  In yards, that's \_\_\_\_  In inches, that's \_\_\_\_  In centimeters, that's \_\_\_\_ |

**Part #3**

1.Create a class called “Assignment02c”

2.Solve the following problem: Write a program that asks the user to enter a point in the form of an X and Y coordinate. Then test to see if the point supplied falls within the circle below. Hint: use the distance formula you calculated for Part 1 to test to see how far the given point is from the origin. Here’s a sample running of this program:

|  |
| --- |
| Enter an X position: 2  Enter a Y position: 2  Point (2,2) is inside the circle! |

**Part #4**

1.Create a class called “Assignment02d”

2.Solve the following problem: Suppose you save $100 each month into a savings account with an annual interest rate of 5%.

Write a program that prompts the user to enter in a monthly savings amount (100 in this case, but it could be more or less) and display the account value after the sixth month.

|  |
| --- |
| <HINT> The monthly interest rate is:0.05/12 = 0.00417  After the first month the value in the account becomes  100 \* (1 + 0.00417) = 100.417  After the second month, the value in the account becomes:  (100 + 100.417) \* (1 + 0.00417) = 201.252  And after the third month the value in the account becomes:  (100 + 201.252) \* (1 + 0.00417) = 302.507 |

**Part #5**

1.Create a class called “Assignment02e”

2.Write a program that constructs a random 3 character word using the uppercase characters from the ASCII encoding set (ASCII characters ‘A’ through ‘Z’). For example, here is some sample output that could be produced by your program:

|  |
| --- |
| // running #1  AXR  // running #2  ZQD  // running #3  AXE |

3.Do not use any random number generation methods such as Math.random(). Instead, you should use the System.currentTimeMillis() method to generate your 3 characters.

**Selection Statements & Loops**

**Part #1**

1.Create a class called “Assignment03a”

2.Write a currency conversion program that converts between US dollars and Euros.

A.Begin by prompting the user to enter the exchange rate from dollars to Euros (i.e. 1 dollar = 0.76 Euros).

B.Next ask the user if they want to convert from Dollars to euros (1) or Euros to Dollars (2)

C.Finally ask them to enter in a currency value. Perform the conversion according to the values they entered. Here’s an example running of the program:

|  |
| --- |
| Enter the exchange rate from dollars to Euros: 0.7623  Enter 1 to convert Dollars->Euros, 2 to convert Euros->Dollars: 1  Enter the Dollar amount: 100  $100.0 is 76.23 Euros |

**Part #2**

1.Create a class called “Assignment03b”

2.Write a program that displays the ASCII table from characters ! through ~. Refer to the ASCII table as necessary to determine your starting and ending points. Display 10 characters per line. Your output should look like the following:

|  |
| --- |
| !"#$%&'()\*  +,-./01234  56789:;<=>  ?@ABCDEFGH  IJKLMNOPQR  STUVWXYZ[\  ]^\_`abcdef  ghijklmnop  qrstuvwxyz  {|}~ |

**Part #3**

1. Create a class called “Assignment03c”

2. Skee Ball is an arcade game that lets players roll a ball up an angled surface and into a series of “hoops”. Players earn points based on which hoop their ball falls into. See the image below to see what the game looks like in real life if you haven’t had the chance to play it yourself.

3. For this program you will be writing a Monte Carlo simulation that will simulate the rolling of 1,000,000 balls into a virtual Skee Ball playing area. To do this you can repeatedly generate a random coordinate within the Skee Ball area and test to see which hoop (if any) the ball falls into. Note that the playing field is 48″ wide by 60″ high, so you will be generating random coordinates within this range as part of your simulation.

4. When you are finished you should calculate the % chance of a ball falling into a particular hoop. Refer to the schematic below for the exact measurements of the Skee Ball board. Hint: Start off by simulating just one ball toss. Once you are confident that you have designed an effective algorithm you can scale up and place your code inside a loop. Here’s a sample running of the program:

|  |  |
| --- | --- |
| Total hits: 1000000  Hits Missed: 628848 (62.88 %)  Hits 40: 17438 (1.74 %)  Hits 30: 27178 (2.72 %)  Hits 20: 39386 (3.94 %)  Hits 10: 287150 (28.72 %) |  |

**Part #4**

1. Create a class called “Assignment03d”

2. Write a program using nested loops to find all the “perfect” numbers between 1 and 10,000. A “perfect” number is equal to the sum of its divisors, excluding itself. For example, 6 is a perfect number because 6 = 3 + 2 + 1. These numbers are quite rare – there are only of them 4 less than 10,000. Your program does not need to accept any input from the user – simply solve the problem and hard code your variables as necessary.

**Part #5**

To "capitalize" a string means to change the first letter of each word in the string to upper case (if it is not already upper case). For example, a capitalized version of "Now is the time to act!" is "Now Is The Time To Act!". Write a subroutine named printCapitalized that will print a capitalized version of a string to standard output. The string to be printed should be a parameter to the subroutine. Test your subroutine with a main() routine that gets a line of input from the user and applies the subroutine to it.

Note that a letter is the first letter of a word if it is not immediately preceded in the string by another letter. Recall that there is a standard boolean-valued function Character.isLetter(char) that can be used to test whether its parameter is a letter. There is another standard char-valued function, Character.toUpperCase(char), that returns a capitalized version of the single character passed to it as a parameter. That is, if the parameter is a letter, it returns the upper-case version. If the parameter is not a letter, it just returns a copy of the parameter.

**Part #6**

Write a subroutine named "stars" that will output a line of stars to a console. (A star is the character "\*".) The number of stars should be given as a parameter to the subroutine. Use a for loop. For example, the command "stars(20)" would output

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Part #7**

Write a main() routine that uses the subroutine that you wrote for Question 7 to output 10 lines of stars with 1 star in the first line, 2 stars in the second line, and so on, as shown below.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

**Part #8**

Write a function named countChars that has a String and a char as parameters. The function should count the number of times the character occurs in the string, and it should return the result as the value of the function.

**Part #9**

Write a subroutine with three parameters of type int. The subroutine should determine which of its parameters is smallest. The value of the smallest parameter should be returned as the value of the subroutine.

**Methods**

**Part #1**

1. Create a class called “Assignment04a”

2. Write a method called “messUp” that accepts in integer and does the following:

◾If the integer supplied is a multiple of 5, divide it by 7 (integer division)

◾If the integer supplied is even, double it

◾If the integer supplied is odd, triple it

3. Return the modified integer to the calling method. Note that some numbers (i.e. 5 ) qualify as both odd and a multiple of 5 – therefore those numbers need to be divided by seven and then tripled before they are returned. Here’s a method header to get you started:

public static int messUp(int number)

4. Set up a loop in your main method that iterates from 1 to 99 and prints the following output (abbreviated).

**Part #2**

1. Create a class called “Assignment04b”

2. Write a method that prints a sequence of characters. By calling this method you can specific the start character, the end character and the number of characters per line. Here’s a method header ot get you started:

public static void printChars(char start, char end, int numPerLine)

3. Next, write some code in your main method that asks the user to enter in a start character, and end character and a number of characters per line. Note that there is no “nextChar()” method on the Scanner class. Therefore you will need to read in the character as a String and then convert it to a character. Use the charAt() method on the String object to extract the first character in the String.

4. Ensure that the start character is less than the end character. If it is not you should display an error to the user and not continue.

5. Finally, display the desired output. Here’s a sample running of your program:

|  |
| --- |
| Start character: a  End character: z  Number per line: 5  abcde  fghij  klmno  pqrst  uvwxy  z |

**Part #3**

1.Create a class called “Assignment04c”

2.Write a method that converts milliseconds to hours, minutes and seconds using the following method header:

public static String convertMillis(long millis)

3.The method should return a string formatted as hours:minutes:seconds. For example, running convertMillis(5500) should return the String “00:00:05″ and convertMillis(100000) should return the string “00:01:40″ (note the leading zeros)

4.In your main method write a program that repeatedly prompts the user to enter in a number of milliseconds (as a long) and displays the corresponding hours, minutes and seconds. End the program when the user supplies the value 0. Here’s a sample running of the program:

|  |
| --- |
| Enter a number of milliseconds, 0 to end: 5500  That's 00:00:05  Enter a number of milliseconds, 0 to end: 100000  That's 00:01:40  Enter a number of milliseconds, 0 to end: 0 |

**Part #4**

1.Create a class called “Assignment04d”

2.Begin by writing a method to reverse an integer using the following method header. Note that you can do this using math expressions or via String manipulation – either approach is fine. public static int reverse (int number)

3.Then write a method called isPalindrome that tests to see if a given integer is a palindrome (reads the same forward and backwards). You should call your reverse method from your isPalindrome method to test to see if a number is a palindrome. Use this method header when writing your isPalindrome method. Note that you can assume that the number entered will not end in zero (i.e. the number 1230 will not be tested)

public static boolean isPaindrome(int number)

4.Finally, write a program that continually prompts the user to enter in an integer. Then report to the user if the number is a palindrome. You can end the program when the user enters a 0. Here’s a sample running of your program:

|  |
| --- |
| Enter an integer, 0 to end: 551  551 reversed is 155. Your number is not a palindrome.  Enter an integer, 0 to end: 1221  1221 reversed is 1221. Your number is a palindrome.  Enter an integer, 0 to end: 0 |

**Single Dimensional Arrays**

**Part #1**

1. Create a class called “Assignment05a”

2. Write a series of methods that operate on a single dimensional array to determine the following:

A.The maximum value contained in the array

B.The minimum value contained in the array

C.The average value of all elements in the array

3. Next, write a method that calculates a letter grade based on a double value.

Use the following grade determiniation chart when calculating your grades:

A - 90 +

B - 80 to 90 (up to but not including 90)

C - 70 to 80 (up to but not including 80)

D - 65 - 70 (up to but not including 70)

F - Less than 65

4. Finally, write a program that prompts the user to enter in a total number of grades to score and a series of grade values. Ensure the user enters grades within the specified range (0 – 100) and that the user enters a positive # of grades. If they do not simply re-prompt them until they provide acceptable input. Use your methods to produce output like the following:

|  |
| --- |
| How many grades to score? 5  Enter grade #1: 100  Enter grade #2: 90  Enter grade #3: 85  Enter grade #4: 82  Enter grade #5: 66  Grades for this class are as follows:  Grade #1: 100.0 (A)  Grade #2: 90.0 (A)  Grade #3: 85.0 (B)  Grade #4: 82.0 (B)  Grade #5: 66.0 (D)  Highest grade: 100.0 (A)  Lowest grade: 66.0 (D)  Average grade: 84.6 (B) |

**Part #2**

1. Create a class called “Assignment05b”

2. Write a method that accepts an array of integers. Your method should analyze the supplied array and eliminate all duplicate items. It should then return an array with the duplicates removed. Use the following method header when designing this method:

public static int[] eliminateDuplicates(int[] list)

3. Note that the array you return should be of the correct size (i.e. if you start with an array of 10 items and remove 4 you should return an array of size 6). Hint: it might make things easier if you use a temporary array to store all unique values that you find in your source array. Then “prune” this array down to its correct size once you’ve isolated all of the unique values in the source array.

4. Test your method using the following array:

int[] sourceArray = {1,4,5,4,1,2,3,5,9,7,12,-5,1,4,-1,-5,12,1};

**Part #3**

1. Create a class called “Assignment05c”

2. Write a method that tests to see if two arrays are identical. For purposes of this assignment an identical array is one in which both arrays share the same elements and the same number of elements, though the elements may be ordered differently. Use the following method header:

public static boolean equals(int[] list1, int[] list2)

3.Then write a program that tests the following lists. Hint – it’s sometimes easier to adjust your source arrays so that they are easier to compare with one another before actually beginning the comparison process.

|  |
| --- |
| // test to see if a0 and a1 are the same  int[] a0 = {5,2,5,6,6,1};  int[] a1 = {5,5,2,6,1,6};  // test to see if b0 and b1 are the same  int[] b0 = {5,5,5,6,6,1};  int[] b1 = {5,2,5,6,1,6};  // test to see if c0 and c1 are the same  int[] c0 = {1,2,3,4}  int[] c1 = {1,2,2,3,4} |

**Part #4**

In this assignment you will be writing your own secret encoder and decoder program.

The program should function as follows:

1. Begin by presenting a menu of options to the user. The user should be able to elect to encode a message, decode a message or quit. Repeatedly prompt the user with these options and end the program when the user elects to quit.

2. If the user elects to encode a message you should prompt them to enter in a series of characters. The user should only enter lowercase letters without any spaces. You should perform some data validation to ensure the message is valid – if not you should re-prompt the user for a new message (i.e. if the user enters the message “java 2013″ you should alert them that this is an invalid message and re-prompt them for a new one). You may want to write a method to handle this validation as you will use it in the next step as well.

3. Next, prompt the user for an “encryption key”. This key will also be a String of lowercase characters without any spaces. Ensure the encryption key is valid before continuing.

4. Now that you have an unencrypted message an an encryption key you should use the key to construct an encoded version of the original message. Hint: use the characters in the encryption key to create a “shift index” – this index can be used to “shift” the alphabet a certain number of characters. Here’s an example of how your shift index should work:

|  |
| --- |
| // original alphabet  a,b,c,d,e,f  // Shift Index = 1  f,a,b,c,d,e  // Shift Index = 2  e,f,a,b,c,d |

Note: the encrypted string that you generate should only contain lower case letters.

5. Present the encrypted version of the message to the user. Here’s a sample running of this part of the program:

|  |
| --- |
| Secret Message Encrypt / Decrypt  Enter an option:  (1) to encrypt a message  (2) to decrypt a message  (0) to exit  Option: 1  Enter an unencrypted word: craig  Enter an encryption key: abc  Encoded Message: ixgom |

6. Next, write the decryption portion of the program. In this part of the program you will prompt the user for an encrypted string and an encryption key. Use the algorithm you wrote for the encryption task above to decrypt the message back into its unencrypted form.

7. Hint: You will need to use the “charAt()” and “length()” method on the String class to obtain individual characters from a string and the total length of a string. Also, you should try and make use of methods to divide up the problem into logical chunks – it will make the problem a lot easier to understand and solve.

**Part #5**

Write a program that will read a sequence of positive real numbers entered by the user and will print the same numbers in sorted order from smallest to largest. The user will input a zero to mark the end of the input. Assume that at most 100 positive numbers will be entered.

**Working with Arrays**

**Part #1**

1. Create a class called “Assignment06a”

2. Write a method that returns a random integer between 1 and 54 (inclusive). However, your method should exclude all numbers passed as arguments. For example, if you call your method with the arguments getRandom(1,2,3,4,5) you would only return a random integer between 6 and 54 (excluding 1,2,3,4 and 5 as possible return values). Valid outputs in this example would be 6-54 (inclusive). Here’s the method header to get you started:

public static int getRandom(int... excludeNumbers)

3. Next, write a program to test your method. You should test your method to exclude the numbers 1-10, 11-20, 21-30 and 41-54 (4 runnings of your method). Here’s a sample running: Random # excluding 1-10: 33

Random # excluding 11-20: 34

Random # excluding 21-30: 2

Random # excluding 41-54: 14

**Part #2**

1. Create a class called “Assignment06b”

2. Write a program that asks the user to enter a number of students. Then prompt the user for the name of each student (a String) and their final score in the class (an integer between 0 and 100). You can assume the user will enter in appropriate values (i.e they will enter a valid string and valid integer – you don’t need to check to see if the integer supplied falls inside the correct range). You will want to store these values in two different arrays.

3. Next, sort the students by their final scores and print out the result to the user. Note that you probably don’t want to use the built in sort() method on the Arrays class – doing so will sort your score array but will leave your name array as-is. You will want to sort both arrays in parallel, meaning that if you move an element to a particular spot in one array you will also have to move the corresponding element in the second array to the same location. Take a look at the selection sort code we wrote in class for a hint on how to get started. Here’s a sample running of this program

|  |
| --- |
| How many students? 3  Student #1 name: John  Student #1 score: 95  Student #2 name: Chris  Student #2 score: 76  Student #3 name: Jason  Student #3 score: 97  Course roster sorted by score:  76 - Chris  95 - John  97 – Jason |

**Part #3**

1. Create a class called “Assignment05c”

2. “Plinko” is one of many games of chance that players can participate in during the “Price is Right” game show. In Plinko, the user is presented with a large flat, triangualr board that is punctuated by a series of pegs. Players place a flat disc at the top of the Plinko board and watch as it falls, hitting pegs along the way. Each time their disc hits a peg it has a 50% chance of going left or right. Players win prizes based on the slot the disc ends in at the bottom of the board.

3. Write a program that simulates Plinko. Your program should prompt the user to enter the number of games they wish to play (i.e. drop 10 discs) and how many slots are the the bottom of their board (i.e. 8). Then simulate the falling of each disc by using a random number to determine if it goes left or right at each junction. Report the path taken by each disc and generate a table that displays your results. Here’s a sample run:

|  |  |
| --- | --- |
| How many discs to drop? 3  How many slots? 8  Disc 1: LRLRLRR - slot 4  Disc 2: RRLLLRR - slot 4  Disc 3: LLRLLRR - slot 3  Report:  Slot 0: no hits  Slot 1: no hits  Slot 2: no hits  Slot 3: 1 hit  Slot 4: 2 hits  Slot 5: no hits  Slot 6: no hits  Slot 7: no hits |  |

4. Hints: A disc will hit n-1 pegs, where n is the number of slots at the bottom of the board. You can calculate which slot a disc lands in by counting up the number of times it fell to the right (for example, in a game with 5 slots your disc might have taken the path LRLL – this would mean that the disc fell into slot #1) — Use an array to keep track of the number of hits to each slot.

**Basic Object Oriented Programming**

**Part #1**

1. Create a class called “Rectangle” that can be used to create customized rectangle objects. Your class should include the following – be sure to comment your class appropriately:

A.A double data field named width that specifies the width of the rectangle.

B.A double data field named height that specifies the height of the rectangle.

C.A double data field named xPosition that specifies the x position of the rectangle.

D.A double data field named yPosition that specifies the y position of the rectangle.

E.A constructor that creates a rectangle with a specified width, height, x and y (i.e. your constructor should accept four double values and use those when setting up a new instance of the Rectangle class)

F.A method named getArea() that returns the area of the rectangle.

G.A method named getPerimeter() that returns the perimeter of the rectangle.

2. Next, create a new class called “Assignment07a”. Do the following in this class:

A.Create an instance of Rectangle with width = 10, height = 10, x = -5, y = 5.

B.Create another instance of Rectangle with width = 10, height = 20, x = -5, y = 10.

C.Display the width, height, xPosition, yPosition, perimeter and area of each of your rectangles, like this:

|  |
| --- |
| Rectangle #1  width & height = 10.0 x 10.0  coordinates = -5.0 , 5.0  area: 100.0  perimeter: 40.0  Rectangle #2  width & height = 10.0 x 20.0  coordinates = -5.0 , 10.0  area: 200.0  perimeter: 60.0 |

**Part #2**

1.Create a new class called “City” that can be used to keep track of location information for a given city. Your class should include the following – be sure to comment your class appropriately:

A.String name

B.double lon (for longitude)

C.double lat (for latitude)

D.A constructor that accepts a name, lon and lat value and stores them in the instance variables for the object

E.A method that reports the current position of a city. Here’s a method header to get you started: public void report()

F.A method that computes the distance from the lon and lat of one city to the lon and lat of another city. Use the standard distance formula to compute this value (let’s pretend that the cities lie on a flat plane and not on a sphere!) Here’s a method header to get you started: public double distanceFrom(City otherCity)

2.Create a new class called “Assignment07b”. Do the following in this class:

A.Prompt the user to enter in a number of cities (i.e. How many cities do you want to create?)

B.Next, ask the user to enter in the name, lon and lat for each city. Note that you will probably need to use two scanners since you are asking for both String and double data. Create a new City object and store it in an array that is designed to hold objects of type City (i.e. City[] myCities)

C.Iterate through your array of Cities and ask each city to report its position.

D.Iterate through your array of Cities and compute the distance from each city to each other city. Ensure that you do not calculate the distance from a given city back to itself (i.e. no need to compute distance between NYC and NYC – the result will be zero) — Here’s a sample running of your program.

|  |
| --- |
| How many cities? 3  City # 1  Enter name: NYC  Enter longitude: 50  Enter latitude: 75  City # 2  Enter name: Chicago  Enter longitude: 25  Enter latitude: 10  City # 3  Enter name: LA  Enter longitude: 0  Enter latitude: 50  --------  City: NYC is at: 50.0, 75.0  City: Chicago is at: 25.0, 10.0  City: LA is at: 0.0, 50.0  NYC is 69.6419413859206 units away from Chicago  NYC is 55.90169943749474 units away from LA  Chicago is 69.6419413859206 units away from NYC  Chicago is 47.16990566028302 units away from LA  LA is 55.90169943749474 units away from NYC  LA is 47.16990566028302 units away from Chicago |

**Part #3**

1. Design a new class called “CodeTimer”. This class should contain the following data fields: A.A private long value named startTime

B.A private long value named endTime

C.Get methods for both startTime and endTime (i.e. getStartTime() and getEndTime()

D.A no argument constructor that initializes startTime with the current time (hint: use System.currentTimeMillis() )

E.A method named start() that resets that startTime to the current time

F.A method named stop() that sets the endTime to the current time

G.A method named getElapsedTime() that returns the elapsed time between startTime and endTime

2. Next, create a new class called “Assignment07c”. This class will be used to compare the performance of the Selection Sort algorithm to the built in Java array sort algoritm. Here’s how your program should work: A.Prompt the user to enter in a number of items to sort (an integer)

B.Create two arrays that are of the specified size. Fill both arrays with random numbers between 0 and the number specified (i.e. if the user enters 1000 you should create two arrays with 1000 elements each. Each array should contain a random number between 0 and 1000) — note that for comparison purposes each array should contain the SAME random number in the same position (i.e. if you put the random # 56 into element #1 in one array you should also put 56 in the second array at the same position)

C.Create an instance of your CodeTimer class and measure how long it takes to create the arrays. Report this to the user.

D.Next, use your CodeTimer instance to track how long it takes to run the selection sort algoritm on one of the arrays. You will need to write this algorithm (we did it in class – just look back to our notes on single dimensional arrays). Report this to the user.

E.Finally, use your CodeTimer instance to track how long it takes to run the built in java.util.Arrays.sort() method on the second array. Report this to the user.

F.Here is a sample running of your program:

|  |
| --- |
| Sort Timer!  Enter a number of random items to sort (i.e. 100000): 100000  Took: 0.012 seconds to create arrays  Took: 2.591 seconds to sort array using the Selection Sort algorithm  Took: 0.027 seconds to sort array using the java.util.Arrays.sort() method |

**Part #4**

1. Write a class whose instances represent a single playing card from a deck of cards. Playing cards have two distinguishing properties: rank and suit. Be sure to keep your solution as you will be asked to rewrite it in Enum Types.

2. Write a class whose instances represents a full deck of cards.

3. Write a small program to test your deck and card classes. The program can be as simple as creating a deck of cards and displaying its cards.

**Graphics, Inheritance and Polymorphism**

Before you start: Download and install the necessary graphics libraries

**Part #1**

1. Create a new sketch of size 500 x 500.

2. Set the background color to grey ( i.e. background(150) )

3. Draw four black squares onto the screen – each square should take up exactly 25% of the available space.

4. Program your squares so that they change to white when the mouse hovers over them.

5. When the mouse is not hovering over a square it should slowly fade to black.

6. Note that you do not need to create any external classes for this project. You can simply build everything inside a single Processing class. But if you wanted to design a class for your Square object you certainly can do so!

7. Hints:

A.Start by drawing 4 squares

B.Use variables to store the colors of your squares. These should be instance variables and not local variables.

C.Most of your code will be in the draw() method.

D.You do not absolutely need to use instance methods for this project, but you could if you wanted to.

**Part #2**

1. Create a new class called Account that contains the following. Note that your Account class should be stored inside a package called “account”.

A. An account number (protected integer)

B. A balance (protected double)

C. The following methods:

public Account(double bal) // constructor

public boolean deposit(double amount)

public double getBalance()

D. Upon construction you should generate a unique account number for the newly created account. Use static variables / methods on the Account class to do this.

E. Next create a class named CheckingAccount that extends your Account class. CheckingAccount should be organized in the “account” package along with the Account class. Do not duplicate the methods already written in the Account class – they will be automatically inherited once you extend the class. This account should contain a constructor that calls the superclass constructor using the super() statement. It should also implement a withdraw method that allows the user to take out money from their account. Checking accounts at our bank come with overdraft protection, so the user can take out up to $100.00 beyond their balance (i.e. they can dip down to -$100.00).

F. Next create a class named SavingsAccount that extends your Account class. Do not duplicate the methods already written in the Account class – they will be automatically inherited once you extend the class. This account should contain a constructor that calls the superclass constructor using the super() statement. It should also implement a withdraw method that allows the user to take out money from their account. Savings accounts at our bank do not come with overdraft protection and the user can only withdraw an amount up to their balance.

**Part #3**

First, create a class that defines a student. This class should contain at least two instance variables for identification number and GPA. It should also contain two constructors a default one and one that accepts two values for the instance variables. In addition to set and get methods, the class should override the toString method from the Object class.

Next, create a class that defines a college student. To do this, the class should inherit the above class. In addition to the inherited instance variables, this class should contain at least one more instance variable that defines the college student’s major. This class should also contain a default constructor and one that accepts three values for the instance variables. The first statement in this constructor should call a constructor in the superclass. This class should also override the toString method.

**Part #4**

The RGBColorChooser applet lets the user set the red, green, and blue levels in a color by manipulating scroll bars. Something like this could make a useful custom component. Such a component could be included in a program to allow the user to specify a drawing color, for example. Rewrite the RGBColorChooser as a component. Make it a subclass of Panel instead of Applet. Instead of doing the initialization in an init() method, you'll have to do it in a constructor. The component should have a method, getColor(), that returns the color currently displayed on the component. It should also have a method, setColor(Color c), to set the color to a specified value. Both these methods would be useful to a program that uses your component.

In order to write the setColor(Color c) method, you need to know that if c is a variable of type Color, then c.getRed() is a function that returns an integer in the range 0 to 255 that gives the red level of the color. Similarly, the functions c.getGreen() and c.getBlue() return the blue and green components.

Test your component by using it in a simple applet that sets the component to a random color when the user clicks on a button

**Part #5**

Turn your applet from the previous exercise into a stand-alone application that runs in a Frame. This is not an easy exercise, since the material on frames in Section 7.7 is sort of sketchy. The information is there if you read carefully. (But I won't think too badly of you if you just look at the solution.)

As another improvement, you can add an "Undo" button. When the user clicks on the "Undo" button, the previous drawing operation will be undone. This just means returning to the image as it was before the drawing operation took place. This is easy to implement, as long as we allow just one operation to be undone. When the off-screen canvas, OSC, is created, make a second off-screen canvas, undoBuffer, of the same size. Before starting any drawing operation, copy the image from OSC to undoBuffer. You can do this with the commands

Graphics undoGr = undoBuffer.getGraphics();

undoGr.drawImage(OSC, 0, 0, null);

When the user clicks "Undo", just swap the values of OSC and undoBuffer and repaint. The previous image will appear on the screen. Clicking on "Undo" again will "undo the undo".

Here is a button that opens the paint program in its own window. (You don't have to write an applet like this one. Just open the frame in the program's main() routine.)

**ArrayLists and Exceptions**

**Part #1**

For this assignment you will be creating a movie database to help keep track of inventory for a Netflix-style online retailer.

1.Begin by writing a class called Movie that contains the following public fields:

A.Title (String)

B.Year Released (Integer)

C.Rating (Double)

D.Quantity (Integer) – defaults to 1 upon construction

2.Next, create a class called Inventory that organizes an arbitrary number of movies. Your class should do the following: A.Initialize an ArrayList of type Movie upon construction

B.Contain a method “add” that adds a movie to the database. Add takes three arguments – a title (String), rating (Double) and a year released (Integer). The “add” method will add the movie in question to the ArrayList if the movie does not already exist in the database. If it does it will simply increase the quantity of the movie. Note that you can use the title and year released as “keys” to determine if a movie exists in the database (i.e. there could be many different Batman movies in the database, but there will only be one Batman released in a given year). Hint: use the “equals” method on the String class to compare if two Strings are equivalent.

C.Contain a method “remove” that removes a movie from the database. Remove accepts the title of the movie and a year value. If the movie already exists in the database its quantity is decreased by one. If the quantity falls below 1 then the movie should be removed completely from the database.

D.Contain a method “toString” that prints out a formatted table that contains all movies in the database. Use the String.format method to format your movie info so that everything lines up nicely.

3.Run your program using the following tester class – expected output is below the class code:

|  |
| --- |
| Title Year Rating Quantity  Happy Gilmore 1996 3.5 1  Star Wars 1977 4.8 1  Return of the Jedi 1983 3.9 2  The Nightmare Before Christmas 1993 5.0 1  Title Year Rating Quantity  Happy Gilmore 1996 3.5 1  Star Wars 1977 4.8 1  Return of the Jedi 1983 3.9 1  The Nightmare Before Christmas 1993 5.0 1  Title Year Rating Quantity  Happy Gilmore 1996 3.5 1  Star Wars 1977 4.8 1  Return of the Jedi 1983 3.9 1 |

**Part #2**

Write a program that calculates the homework average for a student.

1.Begin by prompting the user for 10 homework scores. Homework scores will always be represented by an integer. The easiest way to do this would be to use a Scanner object to get a String value from the user and then parse it into an integer, like this: int number = Integer.parseInt(numberScanner.nextLine());

2.You cannot be assured that the user will give you a valid integer. Provide a safeguard to prevent this issue from crashing your program and re-prompt the user to enter in a new value.

3.At the end of your program display the total points earned and the average score ( / scores by 10). Here’s a sample run of the program:

|  |
| --- |
| Score #1 = 10  Score #2 = 12  Score #3 = 15  Score #4 = apple  Uh, that's not an integer. Try again.  Score #4 = pear  Uh, that's not an integer. Try again.  Score #4 = 12  Score #5 = 13  Score #6 = 11  Score #7 = 10  Score #8 = 9  Score #9 = 12  Score #10 = 5  Total points: 109  Average score: 10 |

**Part #3**

Implement the following additional methods into the Inventory class that you wrote for Part 1 above:

1.public Movie[] getMoviesByRating() – returns all movies ordered by their rating. Do not return the object(s) that are stored in the ArrayList (i.e. you want to prevent the user from making changes to the original copy).

2.public Movie[] getMoviesByRating(double minimum) – returns all movies ordered by their rating that have a rating greater than or equal to a minimum value. Do not return the object(s) that are stored in the ArrayList.

3.public Movie[] getMoviesByYear(int year) – returns all movies that were released during a given year. Do not return the object(s) that are stored in the ArrayList.

4.public String sortByTitle() – sorts the ArrayList by title and returns its value as a formatted String.

5.public String sortByYear() – sorts the ArrayList by year released and returns its value as a formatted String.

**Part #4**

Write a subroutine that prints out a 3N+1 sequence starting from a given integer, N. The starting value should be a parameter to the subroutine. If the parameter is less than or equal to zero, throw an IllegalArgumentException. If the number in the sequence becomes too large to be represented as a value of type int, throw an ArithmeticException.

**Working with Files and Interfaces**

**Part #1**

Write a program that creates a data file with 1000 lines of randomzied student enrollment records. Name your file “enrollment.txt”. Each line should represent a student in the following format: First\_Name,Last\_Name,Class\_Year,GPA

First\_Name and Last\_Name can be generated by appending the record number to the end of the Strings “First\_Name” and “Last\_Name”, like this:

First\_Name1,Last\_Name1

First\_Name2,Last\_Name2

...

First\_Name1000,Last\_Name1000

Class years should be randomized to one of the possible states:

Fr / So / Jr / Sr / Gr

GPA should be a floating point number between 0.0 and 4.0, formatted to 1 decimal place.

Here’s a sample file of 5 lines (yours will have to be 995 lines longer than this one!)

First\_Name1,Last\_Name1,Fr,3.5

First\_Name2,Last\_Name2,Jr,0.7

First\_Name3,Last\_Name3,Sr,3.9

First\_Name4,Last\_Name4,Gr,1.5

First\_Name5,Last\_Name5,Fr,3.1

**Part #2**

|  |
| --- |
| FirstName1 LastName1 assistant 79174.73  FirstName2 LastName2 associate 70817.75  FirstName3 LastName3 associate 69619.0  FirstName4 LastName4 full 116992.43  FirstName5 LastName5 full 116761.76  FirstName6 LastName6 full 123743.86  FirstName7 LastName7 assistant 70071.81  FirstName8 LastName8 assistant 67605.92  FirstName9 LastName9 associate 65534.42  FirstName10 LastName10 full 88528.43  FirstName11 LastName11 associate 74674.29  FirstName12 LastName12 associate 104063.63  FirstName13 LastName13 full 125745.3  FirstName14 LastName14 associate 72078.69  FirstName15 LastName15 associate 79000.61  FirstName16 LastName16 associate 98181.07  FirstName17 LastName17 full 83329.54  FirstName18 LastName18 associate 91046.13  FirstName19 LastName19 full 122418.96  FirstName20 LastName20 assistant 59408.67  ...... |

Write a program that interfaces with this data file(You have to make!) and determines the following. Hint: when comparing String values remember that the strict equality operator ( == ) will compare the memory addresses of the Strings in question, not their values. Use the equals method (i.e. if (myString.equals(“another string”) )

1.The total salary for all Professors

2.The total salary for all Assistant Professors

3.The total salary for all Associate Professors

4.The total salary for all Full Professors

5.The average salary for all Professors

6.The average salary for all Assistant Professors

7.The average salary for all Associate Professors

8.The average salary for all Full Professors

Here’s the solution data set so you can check your answer:

Rank # Total Salary Average Salary

Assistant 307 20246511.91 65949.55

Associate 344 28844146.58 83849.26

Full 349 35678051.41 102229.37

--------------------------------------------------------------------------

All 1000 84768709.90 84768.71

**Part #3**

Find all congressional representatives for the selected state. Representative designation is also stored in the “District” column and encoded as follows: StateCode + 2 digit integer code (i.e. 01 for district 01, 02 for district 02, etc). Each state has a different number of representatives. You will want to find each of these and add them to an ArrayList. Then print out the representative’s name, district, phone number and URL, if available. Print these in ascending order (sorted by the last name of the representative)

BIG HINT #1: Write a simple class to store representatives and store them in an ArrayList. Then implement the Comparable interface on this class so that representatives can be compared against one another. You can then use “Collections.sort( myArrayList );” to sort your ArrayList using the logic you outlined in your compareTo method.

BIG HINT #2: Implementing the Comparable interface for this problem is easier than it sounds. The String class already implements the Comparable interface, so you are allowed to do something like this:

int firstNameComparison = this.firstName.compareTo( otherRep.firstName );

to see how two Strings compare to one another (+1 = the source String is bigger, -1 = the source String is smaller, 0 = the Strings are identical)

**Part #4**

Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "java LineCounts file1.txt file2.txt file3.txt". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.

**Part #5**

For this exercise, you will write a network server program. The program is a simple file server that makes a collection of files available for transmission to clients. When the server starts up, it needs to know the name of the directory that contains the collection of files. This information can be provided as a command-line argument. You can assume that the directory contains only regular files (that is, it does not contain any sub-directories). You can also assume that all the files are text files.

When a client connects to the server, the server first reads a one-line command from the client. The command can be the string "index". In this case, the server responds by sending a list of names of all the files that are available on the server. Or the command can be of the form "get <file>", where <file> is a file name. The server checks whether the requested file actually exists. If so, it first sends the word "ok" as a message to the client. Then it sends the contents of the file and closes the connection. Otherwise, it sends the word "error" to the client and closes the connection.

Ideally, your server should start a separate thread to handle each connection request. However, if you don't want to deal with threads you can just call a subroutine to handle the request.

**Part #6**

Write a complete program that will display the first ten lines from a text file. The lines should be written to standard output, System.out. The file name is given as the command-line argument args[0]. You can assume that the file contains at least ten lines. Don't bother to make the program robust.