**Exercises – Methods** Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_\_\_\_\_\_\_

# Consider the following methods:

# public static void **countDown**(int n)

# {

# for(int i=n; i>=1; i--)

# System.out.println(i);

# }

# public static void **liftOff**()

# {

# System.out.println(“We have ignition”);

# System.out.println(“and lift-off!”);

# }

# public static double **motorBurnTime**(double impulse, double thrust)

# {

# return impulse/thrust;

# }

# public static double **acceleration**()

# {

# return 9.81;

# }

# In a driver program’s main method, write the code to call the methods above:

# Call the **countDown** method such that it starts at the value 10:

# Call the **liftOff** method:

# Display the results of the **motorBurnTime** method when the impulse is 18 and the thrust is 4.5:

# Create a variable called x and store into it the value returned from **motorBurnTime** when the impulse is 22.4 and the thrust is 8.5:

# Write a method that returns the *Force* of gravity when given the *mass* of an object as an argument. Call the **acceleration** method to get the value of g in *Force* = *mass*\*g.

# Write a method that displays to the screen the name of your favorite president on one line, followed my your favorite band on the next and your favorite food on the last.

1. Write a method that returns the value of e with an accuracy of 5 digits to the right of the decimal. (You don’t need to calculate it – feel free to google it and just return a value).
2. Write a method that, when given the appropriate dimensions of a triangle, returns its perimeter.

# Exercises Object orientation Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**public class** Coordinate Period\_\_\_\_\_\_\_\_\_\_\_\_

{

**private int** x; //data fields

**private int** y;

*//default constructor*

**public** Coordinate()

{

x = 0;

y = 0;

}

*//constructor //mutator method*

**public** Coordinate(**int** nX, **int** nY) **public void** setX(**int** newX)

{ {

x = nX; x = newX;

y = nY; }

}

*//mutator method //mutator method*

**public void** advance() **public void** setY(**int** newY)

{ {

x++; y = newY;

y++; }

}

*//mutator method //accessor method*

**public void** mult(**int** scalar) **public int** getX()

{ {

x = x \* scalar; **return** x;

y = y \* scalar; }

}

*//lets you SOP a coordinate object //accessor method*

**public** String toString() **public int** getY()

{ {

**return** (“” + x + “, “ + y); **return** y;

} }

}

1. Assuming the code would compile, what would be the output of the following:

Coordinate one = **new** Coordinate();

Coordinate two = **new** Coordinate(3, -4);

Coordinate three = **new** Coordinate(1,1);

one.advance();

two.mult(10);

three.mult(two.x);

System.out.println(one + “\t“ + two + “\t” + three);

1. Write another member method called retreat() that subtracts one from each data member:
2. Find the syntax error in #1:

**public class** Cat

{

**private** String name;

**private** double weight;

**public** Cat () *//default constructor*

{

name = "no name";

weight = 0;

}

**public** Cat (String n, **double** w) *//constructor*

{

name = n;

weight = w;

}

}

//…in a driver program’s main method is this…

Cat x = **new** Cat();

4. Write the code that would make another cat with the name “Pumkin” and with a weight of 8.3:

5. Overload the toString() method in the cat definition that will return the cat’s information as a String:

6. Modify the class declaration to include the number of lives that the cat has left, and modify the constructors to account for this:

7. Write a class definition that represents a Card that stores its value (1-13 for ace through king), its suit ("H", "C", "D", "S") and whether it has been drawn from the deck or not (true or false):

8.Write the code to create a card and initialize it to the Ace of Spades:

**Exercises – Defining Objects** Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_\_\_\_\_\_\_

1) Define a class for a Vehicle with the following properties:

Data fields: name, # passengers

Methods: 2 argument constructor, toString, getName(), getPassengers(), setName()

2) Define a class called Point with the following properties:

Data fields: x, y and z position in 3-D space

Methods: 3 argument constructor, toString (overloaded to show position),

3 argument mutator to change the Points position, 3 accessor methods3) Define your own object definition for any type of object you chose. Pick logical data fields and methods.

|  |  |
| --- | --- |
| **String Method Summary** | |
|  | Here are some useful methods that are built into String objects |
|  |  |
| int | [**compareTo**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#compareTo(java.lang.String))([String](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html) anotherString)            Compares two strings alphabetically.  i.e. if(word1.compareTo(word2) < 0) will see if word1 < word2  word1.compareTo(word2) == 0 means they are the same  word1.compareTo(word2) > 0 means word1 > word2 |
|  |  |
| boolean | [**equals**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#equals(java.lang.Object))([Object](file:///C:\JAVA2005\APIs\sun%20api\java\lang\Object.html) anObject)            Compares this string to the specified object.  Ex. if(word.equals(“yes”) || word.equals(“Yes”) || word.equals(“YES”)) |
| int | [**indexOf**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#indexOf(int))(String sub)           Returns the index within this string of the first occurrence of the specified substring, -1 if not found.  Ex. String word = “Washington”;  word.indexOf(“W”) returns 0  word.indexOf(“ing”) returns 4  word.indexOf(“n”) returns 5  word.indexOf(“zk”) returns -1 |
| int | [**length**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#length())()            Returns the length of this string. int size = word.length(); |
| [String](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html) | [**substring**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#substring(int))(int beginIndex)            Returns a new string that is a substring of this string, from beginIndex all the way to the end.  Ex. String word = “Washington”;  word.substring(7) returns “ton” |
| [String](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html) | [**substring**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#substring(int, int))(int beginIndex, int endIndex)            Returns a new string that is a substring of this string, including beginIndex but stopping before (and thus not including) endIndex.  Ex. String word = “Washington”;  word.substring(1,4) returns “ash” |
| [String](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html) | [**toLowerCase**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#toLowerCase(java.util.Locale))()  Returns a new, lower case version of the String.  Ex. word = word.toLowerCase(); |
| [String](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html) | [**toUpperCase**](file:///C:\JAVA2005\APIs\sun%20api\java\lang\String.html#toUpperCase())()            Returns a new version of the string with all upper case characters.  Ex. word = input.next().toUpperCase(); |

**INFO** Conditions with equalities and inequalities:

You can test the value of any primitive (int, double, boolean) using the operators

== (equals), != (not equal to), <, <= (less than or equal to), > and >=.

Examples:

//given an int variable called temp:

if(temp < 32)

System.out.println(“It might snow”);

//given a double variable called x:

if(x != 0)

System.out.println(x + “ is not zero”);

You can test the equality of a String using the method .equals (dot-equals).

Examples:

//given a String called name:

if(name.equals(“Oberle”))

System.out.println(“Do you need another vanity title?”);

//given a String called answer:

if(!ans.equals(“A”) && !ans.equals(“B”))

System.out.println(“That is not a valid option”);

You can test greater than and less than with Strings by using the method .compareTo (dot-compareTo). This method returns a number that will be negative if the String you are calling it from is less than the argument you send it. It will return zero if they are the same and a positive if the argument is greater. The determination of comparison is alphabetic. Strings that start with an ‘A’ are considered less than those that start with a ‘B’.

Examples:

//given two Strings called word1 and word2:

if(word1.compareTo(word2) < 0)

System.out.println(word1 + “ is less than “ + word2);

The condition (word1.equals(word2)) is the same as (word1.compareTo(word2) == 0).

**Strings and Comparisons Exercises** Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Period\_\_\_\_\_\_\_\_\_\_\_\_\_

Consider the following code:

String word = “Scaramouche”;

1. Write the code that would access the String above to write “Scar” out to the screen (using the method substring).
2. Write the code that would write “mouch” out to the screen using substring.
3. Write the code that would write “ara” out to the screen using substring.

Consider the following additional code to the code above, given that word stores the value “Scaramouche”:

int x = word.indexOf(“car”);

int y = word.indexOf(“ouch”);

System.out.println(word.substring(x,y));

1. Show is the output of the code above.

Consider the following code:

String x = new String(“Hayes”);

String y = “Rutherford”;

String z = x + “B“ + y;

System.out.println(z);

int i = z.indexOf(“B“);

System.out.println(z.substring(i+1) + “ “ + z.substring(i, i+1) + “ “ + z.substring(0,i));

1. Show is the output of the code above.
2. Write the code that will take a String called word and if it ends with “ing” will write the word out to the screen without the “ing” on the end.

I.e., If word store the value “hello”, the code would write “hello”.

If word stores the value “jousting”, the code would write “joust”.

(Assume that the word will have a length of at least 4).

7)

int num;

System.out.println(“enter a value”);

num = input.nextInt();

//write the code that will print the message “OK” if num is larger than 100

8)

String word;

System.out.println(“enter a word”);

word = input.nextLine().toLowerCase();

//write the code that will print “that’s the secret word” if the user types in “crap”.

9)

double x, y;

System.out.println(“enter a number”);

x = input.nextDouble();

System.out.println(“enter another number”);

y = input.nextDouble();

//write the code that will add 1 to the value of y only if it is less than or equal to x.

10)

String name1, name2;

System.out.println(“enter a name”);

name1 = input.nextLine();

System.out.println(“enter a name”);

name2 = input.nextLine();

//write the code that will print the name of the person that comes first alphabetically, and

//states that they will be served next. i.e., if the user types in Giger for name1 and Bosch

//for name2, the code will respond with “Bosch will be served next”.