Final Exam Practice Questions

1. **Short Answer Questions** (10 points total)

   (a) Given the following hierarchy:

```
class Alpha { ... }
class Beta extends Alpha { ... }
class Gamma extends Beta { ... }
```

   What order are the constructors for these classes called when a Gamma object is instantiated? (1 point)

   (b) What class is a superclass of every class in Java? Name three methods inherited from that class? If you have an object, `obj`, how would you call those three methods? (2 points)

   (c) Explain what the extends keyword is used for (of course, without re-using the actual word *extends*!). (1 point)

   (d) Write the code to print out if the file, hello.txt, exists (assume that any necessary imports are already included, so there is no need to write them out). Then, open the file to read in and output the 1st line of text. (2 points)

   (e) How many classes can extend a superclass? How many superclasses can a class extend? (1 point)

   (f) Describe two ways in which abstract classes and interfaces differ. (1 point)

   (g) Name 2 wrapper classes and their related primitives. (1 point)

   (h) What is autoboxing? (1 point)

2. **Definitions** – define the following terms (1 or 2 sentences for each), and show an abbreviated code example demonstrating each term (5 or 6 lines of code, with comments where appropriate). Code does not have to be complete / runnable. (10 points)

   (a) **Encapsulation**

   (b) **Inheritance**

   (c) **Polymorphism**

   (d) **Aggregation**

   (e) **Composition**

3. **True or False** (10 points)

(a) True / False – A superclass reference can refer to a subclass object.

(b) True / False – A subclass reference can refer to a superclass object.

(c) True / False – A NullPointerException is a checked exception that must be caught or declared.

(d) True / False – To make a class immutable, all data fields must be private and final.

(e) True / False – A class may only implement a single interface.

(f) True / False – A regular method (not an abstract method) can be contained in an abstract class.

(g) True / False – An abstract class can be used as a data type when declaring a variable

(h) True / False – An abstract method must be nonstatic.

(i) True / False – Not all exceptions are runtime errors.

(j) True / False – The JVM searches for a method's implementation in the object's superclasses first.

4. **What's the output?** Read the program in the 1st column. Write the output of the program in the second column. (10 points)

`public class ConConConstructor {`
`  public static void main(String[] args) {`
`  A a = new A(3);`
`  }`
`}`
`class A extends B {`
`  public A(int t) {`
`  super();`
`  System.out.println("A's constructor!");`
`  }`
`}`
`class B extends C {`
`  public D[] ds;`
`  public B() {`
`  ds = new D[2];`
`  ds[0] = new D();`
`  ds[1] = new D();`
`  System.out.println("B's constructor!");`
`  }`
`}`
`class C {`
`  public C() {System.out.println("C's constructor!");}`
`}`
`class D {`
`  public D() {System.out.println("D's constructor!");}`
`}`	

`public class Foo {`
`  public static void main(String[] args) {`
`  Object o1 = new Baz();`
`  Object o2 = new Qux();`
`  System.out.print(o1);`
`  System.out.print(o2);`
`  }`
`}`
`class Baz extends Qux {`
`  public String toString() {`
`  return "Baz"; }`
`}`
`class Qux {`
`  public String toString() {`
`  return "Qux";`
`  }`
`}`	

`public class Division {`
`  public static void main( String [] args ) {`
`  Scanner in = new Scanner (System.in);`
`  int num1 = 0, num2 = 0;`
`  try{`
`  System.out.println("Dividend and divisor, plz:");`
`  num1 = in.nextInt ();`
`  num2 = in.nextInt ();`
`  System.out.printf ("Quotient is %s", num1/num2);`
`  } catch (ArithmeticException e) {`
`  System.out.println("Can't divide by 0"); }`
`  catch (InputMismatchException e) {`
`  System.out.println("Not an integer!");`
`  }`
`  catch ( Exception e ){`
`  System.out.println(e);`
`  System.out.println("???");`
`  }`
`  in.close();`
`  }`
`}`
	What's the output for the following user input?
	
	5 and 2.5
	
	10 and 5
	
	5 and 0

5. **Finding and fixing bugs.** (10 points)

```
public class ReverseInputBuggy {

  public static void main( String [] args ) {

    Scanner in = new Scanner(input.in);

    int numInts = in.nextInt();

    String strings;

    for (int i = 1; i < numInts; i++) {

      strings[i] = in.next();

    }
    for (int i = numInts; i >= 0; i--) {

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

    }
  }
}
```

The code on the left asks for a number. Based on that number, it will ask for that number of additional input. It will then print out the numbers inputted in reverse order. There are 2 compilation erros, and 2 runtime errors. **Find** and **correct** them.

```
public class ClassesBuggy {
  public static void main(String[] args) {
    MyClass3 m3 = new MyClass3();
  }
}

abstract class MyClass1 {
  public String myMethod() {return "class1";}
  abstract int getNumber() {return 1;}
}

class MyClass2 {
  public String myMethod() {return "class2";}
}

class MyClass3 extends MyClass2, myClass1 {
  @Override
  public int myMethod() {return 3;}

}
```

Find at least two errors in the code to the left (there are more than two). Write a short description of why there's an error next to the error.

6. **Multiple Choice.** There might be more than one correct answer - mark all that apply.

(a) A subclass inherits _____ from its superclass.

i. private methods ii. protected methods iii. public methods iv. static methods

(b) When you implement a method that is defined in a superclass, you _____ the original method.

i. overload ii. override iii. call iv. copy

(c) An ArithmeticException is a subclass of:

i. Object ii. Throwable iii. Exception iv. RuntimeException v. Catchable

(d) Checked exceptions:

i. are not checked by the compiler
ii. must be dealt with in a try catch
iii. must be declared
iv. must be dealt with in a try catch and must be declared
v. must be either dealt with in a try catch or declared

(e) What is the output of the following code?

```
public static void main(String[] args) {
  Object o = new Object();
  String d = (String) o;
  System.out.println(d);
}
```

i. NullPointerException ii. ClassCastException iii. "" iv. d

(f) What code would be used to convert the following variable into a double?

```
String s = 5.0
```

i. (double) s
ii. s.toDouble()
iii. Double.parseDouble(s)
iv. cannot convert a String to a double

(g) This kind of file contains your java source code:

I. .class ii. .java iii. txt iv. .bytecode

(h) If you would like to join strings together without creating new strings every time they are joined, use:

i. Concatenation (s + s)
ii. StringBuilder
iii. StringBuffer
iv. the String constructor (new String(s + s))

(i) What is the contents of args[1] if your commandline arguments (through Eclipse Run Configuration) are: foo bar baz (that is, if you run your file as java MyClass foo bar baz)

i. error ii. foo iii. bar iv. baz

7. **Fill in the missing code**. Finish the implementations for the following... (10 points)

(a) binary search

```
public static int binarySearch(int[] numbers, int n) {

    int start = _____________________________;

    int end = _____________________________;

    while (_____________________________________) {

      int mid = _____________________________;

      if(numbers[mid] == n) {

        ________________________________

      } else if (numbers[mid] > n) {
        end = mid - 1;
      } else {
        start = _____________________________;
      }
    }
    return -1;
  }
}
```

(b) fill in the implementations for the class and constructor below so that the output of the 1st and 2nd columns are both 42.

```
class C {
  public int foo ()
  {
    return 10;
  }
}
class S extends C {

  // finish this class definition









}
class Main {
  public static void main(
      String [] args  {
    S x = new S(42);
    System.out.println(x.foo());
  }
}
```

```
class C {
  private int x;
  public C(int x) {
    this.x = x;
  }
  public int foo () {
    return x;
  }
}
class S extends C {
  public S() {

    // finish this constructor









  }
}

class Main {
  public static void main(
      String [] args ) {
    S x = new S();
    System.out.println(x.foo());
  }
}
```

8. Create a method called **`decode`**. It will translate a string encoded with ROT13, a simple "encryption" algorithm that shifts every letter forward 13 characters. For example, the letter a would be shifted to n. To "decode" the letter n, go back 13 letters to a. Write a method that:

   (a) takes a ROT13 encoded **String as an argument**
   (b) **returns a String** representing the decoded version of the original argument
   (c) example: `decode("njrfbzr cbjre")); // returns: "awesome power"`
   (d) to decode, shift each letter back by 13
   (e) if shift results in going back before the letter, a, continue from the end of the alphabet and go backwards - for example, the letter j would be shifted back to w
   (f) only decode lowercase letters (punctuation, spaces, etc., should remain the same)
   (g) hint: String has a constructor that can take a char Array
   (h) hint: or… StringBuilder/StringBuffer has an append method that can take a char

9. Write the following program:

   (a) **Create a `Rectangle` class**; it should have a width and a height, and it should be able to derive an area and String representation of itself based on width and height (the String representation should be: `"[width]x[height], area:[area]" - "10x2, area:20.00"`)

   (b) **Write a program that asks the user for a width and a height 10 times**
      i. assume that the input is always valid (you don't have to take care of invalid input)
      ii. assume that you're writing code that goes into a main method (you don't have to write the main method header... any imports are magically imported for you)
      iii. **create a `Rectangle` object** for every set of width and height entered by the user
      iv. **print out the rectangle with the largest area**

   (c) Example output (everything after > is user input):

```
Width for rect 1 > 2
Height for rect 1 > 8
Width for rect 2 > 10
…. (continues through rect 10)
The largest rect is: 12x6, area:72.00
```

10. Using the Rectangle class from the previous question, write the following program.

    (a) Read a file called `rectangles.txt`
    (b) Each line in the file is in the format [width]x[height]
    (c) Create and store a new Rectangle object for every line (use whatever method you like)
    (d) Print out all of the rectangles in reverse order

11. **Create three classes**, `Cat`, `Dog`, and `Animal`. **One of the classes must contain an abstract method**. **Base your class definitions on** the the following sample code and its output:

```
Animal[] animals = {
      new Cat("frisky", 7), new Cat("fluffy", 8),
      new Dog("fetch", 10), new Dog("fido", 20)
};

for(Animal animal: animals) {
      System.out.printf("%s - %s\n",
                  animal.getName(), animal.makeNoise());
}
System.out.println(((Cat) animals[0]).lives);
```

Running the code above prints out:

```
frisky - meow
fluffy - meow
fetch - woof
fido - woof
9
```

12. Write a CalendarDate class that has two private data fields day, and month, both ints.

    (a) it should have a default constructor that sets the date to a random day in May (May has 31 days)
    (b) it should have an overloaded constructor that sets the date to a the month and day passed in; if the month is not valid, then make the program crash with an IllegalArgumentException
    (c) an Array of CalendarDate classes should be sortable by using Arrays.sort()... write the code necessary to make this feature work
    (d) passing a CalendarDate Array to Arrays.sort will sort the Array by chronological order (for example, Jan 1 would go before Feb 1)

13. Using the class definition for CalendarDate in the previous question, write:

(a) a method that tests for equality; this method should override the one that is inherited from Object – two dates should be considered equal if their month and date are the same
(b) a method that overrides Object's toString method; it should represent a CalendarDate object as [month]/[date], for example, 5/31
(c) a main method that:
    i. creates an Array of 3 CalendarDate objects, 2 of them random, the third set to January 2nd explicitly
    ii. sorts the Array (don't write your own sort for this, take advantage of the fact that an Array of CalendarDates can be sorted!)
    iii. print sout out the entire sorted Array as a String
    iv. finally, creates a new random date object – checks if it's in the Array, and prints out found if iti exists in the Array

14. For the purpose of this question, assume that you are developing static methods for your own statistics library MyStats (it is similar to the Java's Math library that you have been using throughout the semester).

    (a) Write a public static method threeEqual() for MyStats that takes three int values as arguments and returns true if all three numbers are equal, false otherwise.

    (b) Write a public static method median() for MyStats that takes as an argument an array of sorted integer values and returns the middle value (if there are an even number of elements, return the average of the two middle values). Your method should first verify if the array is sorted, and if it is not it should throw an IllegalArgumentException.

    (c) Give a single Java expression that a user of the MyStats class could write to test whether the medians of three arrays of integer values int[] a, int[] b, and int[] c are all equal.

15. Write the method sumSeries() that given an integer argument n will return as a double the following sum:

    $1/n + 2/(n - 1) + 3/(n - 2) + \ldots + (n - 1)/2 + n/1$

    Do not write a full program with input and output, just the method.

    Hints/Notes:
    (a) You need only a single loop.
    (b) The return value is a double, make sure that as the sum is computed using double division and not integer division.
    (c) You don't need to check for valid values of n, assume it's an integer > 0.

Char	Dec	Char	Dec	Char	Dec	Char	Dec
(nul)	0	(sp)	32	@	64	`	96
(soh)	1	!	33	A	65	a	97
(stx)	2	"	34	B	66	b	98
(etx)	3	#	35	C	67	c	99
(eot)	4	$	36	D	68	d	100
(enq)	5	%	37	E	69	e	101
(ack)	6	&	38	F	70	f	102
(bel)	7	'	39	G	71	g	103
(bs)	8	(	40	H	72	h	104
(ht)	9	)	41	I	73	i	105
(nl)	10	*	42	J	74	j	106
(vt)	11	+	43	K	75	k	107
(np)	12	,	44	L	76	l	108
(cr)	13	-	45	M	77	m	109
(so)	14	.	46	N	78	n	110
(si)	15	/	47	O	79	o	111
(dle)	16	0	48	P	80	p	112
(dc1)	17	1	49	Q	81	q	113
(dc2)	18	2	50	R	82	r	114
(dc3)	19	3	51	S	83	s	115
(dc4)	20	4	52	T	84	t	116
(nak)	21	5	53	U	85	u	117
(syn)	22	6	54	V	86	v	118
(etb)	23	7	55	W	87	w	119
(can)	24	8	56	X	88	x	120
(em)	25	9	57	Y	89	y	121
(sub)	26	:	58	Z	90	z	122
(esc)	27	;	59	[	91	{	123
(fs)	28	<	60	\	92	|	124
(gs)	29	=	61	]	93	}	125
(rs)	30	>	62	^	94	~	126
(us)	31	?	63	_	95	(del)	127