Midterm 2 - SAMPLE

CSC-121 Fall 2025

Name: _____

Multiple Choice Questions: (12 points)

Please circle out or mark the left side of the correct answers.

1. Methods are useful because they _____.

    a. …are reusable, cutting down duplicate code.
    b. …make code more modular and readable by placing features in dedicated blocks with descriptive names.
    c. …can act as a public interface to interact with private elements of classes.
    d. …can perform the same action on generalized input parameters without requiring separate code.
    e. …can be overloaded to take in different data types to perform similar actions.
    f. all of these
    g. none of these

2. To create a new array in memory, use this keyword:

    a. construct
    b. new
    c. array
    d. object

3. Which of the following would correctly create an array for holding 5 ints?

    a. new int[] nums = int[5];
    b. int[] num = new int[5];
    c. int[5] num = new int[];
    d. int num[] = new int[5];
    e. All of the above
    f. a and b
    g. c and d
    h. a and c
    i. b and d

4. What does the following method return?

```
public int upToTen(int n)
{
      return (n % 10);
}
```

   a. An even number.
   b. An odd number.
   c. A value between 0 and 9.
   d. True or False.

5. Given the following method, which of these method calls is valid?

```
public static void sumNums(int num1, double num2)
{
      double sum;
      sum = num1 + num2;
      System.out.println("The sum is " + product);
}
```

   a. `sumNums("5", "4.0");`
   b. `sumNums(10.0, 4.6);`
   c. `sumNums(10, 4.5);`
   d. `sumNums('a', 5.2f);`
   e. `sumNums(3.3f, 55);`
   f. `sumNums(true, 60);`
   g. a and b
   h. c and d
   i. e and f

6. This keyword allows code to exit out of a loop early.

   a. end
   b. break
   c. exit
   d. stop

True/ False questions: (12 points)

Select **T** only if the *entire statement* is true. If *any part* of the statement is false, select **F**.

1. **T** or **F** : Like arrays, ArrayLists can hold any Java datatype.

2. **T** or **F** : Variables defined in a main method by default have scope encompassing the entire program.

3. **T** or **F** : Java requires that you must write a constructor for all classes.

4. **T** or **F** : Java methods can change the state of an object created outside of the method if the method has a reference to that object passed in as an input parameter.

5. **T** or **F** : Java methods can return multiple values by using a return statement with commas separating each value.

6. **T** or **F** : The static keyword for a method means that the method can be called without a reference to a specific instance of that class.

Short Answer: (8 points)

Write a few sentences to answer each question. Give examples if appropriate.

1. Why might a class have more than one constructor, and how might it be useful?

2. What is the point of a "void" method - a method that does not return any value? How can void methods still be useful?

3. What is the purpose of the angle brackets <> when defining an ArrayList? Give an example.

4. In the following code, what is printed? Explain.

```
public class ChangeUp
{
	public static void main(String[] args)
	{
		int i = 1;
		double[] j = {2.0, 3.14};
		System.out.println(i + " " + j[0]);
		changeUs(i, j);
		System.out.println(i + " " + j[0]);
	}

	public static void changeUs(int a, double[] b)
	{
		b[0] = b[0] + a;
		a = 0;
		System.out.println(a + " " + b[0]);
	}
}
```

Semi-Programming: 28 points

Write short code segments for each question. You do not need to write an entire program, just a snippet that would perform the requested action if it were part of a full program.

1. (7 points)

Write a method named sumThrees that takes in an array of positive integers and returns the sum of all of the numbers in the array that are a multiple of 3.

Example:

```
int[] nums = {2, 1, 3, 4, 6, 5, 9, 8, 7};
System.out.println(sumThrees(nums));  // 3+6+9=18, Prints: "18"
```

2. (7 points)

Write a method named sumColumn that takes a 2D array of integers (`int[][]`) and an integer column index. Your method should return the sum of all of the elements from that column of your array. If a particular row is not long enough to have an entry for the given column index, count that row's column value as 0.

Example:

```
int[][] array2d = {{1, 2}, {3, 4, 5}, {6, 7, 8, 9}};
int column = 2;
System.out.print(sumColumn(array2d, column)); // 0+5+8=13, Prints "13"
```

3. (7 points)

Write a method named mixto that takes in as input parameters two int arrays of the same length named a and b, and outputs a new int array that has the elements of both arrays mixed together. The order of the new array should be the first element of a followed by the first element of b, then the second element of a followed by the second element of b, etc.

Example:

```
int[] a = {1, 2, 3};
int[] b = {4, 5, 6};
int[] c = mixto(a, b);
for (int i = 0; i < c.length; ++i) {
    System.out.print(c[i] + " ");
}
// Prints: "1 4 2 5 3 6 "
```

4. (7 points)

Write a method called insertName that takes two arguments: An ArrayList<String> of single-word names in alphabetical order, and a new name to add to the list. Your method should insert the new name into the ArrayList in alphabetical order. You may ignore uppercase/lowercase differences for purposes of alphabetical sorting.

HINT: The String class has methods compareTo and compareToIgnoreCase which might be useful here. Below is a snippet from the official Java Documentation:

compareTo / compareToIgnoreCase

`public int compareTo(String anotherString)`

Compares two strings lexicographically … based on the Unicode value of each character in the strings. …The result is a negative integer if this String object lexicographically precedes the argument string. The result is a positive integer if this String object lexicographically follows the argument string. The result is zero if the strings are equal; compareTo returns 0 exactly when the equals(Object) method would return true.

Example:

```
ArrayList<String> names = new ArrayList<>();
insertName(names, "Pinky");
insertName(names, "Blinky");
insertName(names, "Inky");
insertName(names, "Clyde");
System.out.println(names);  // Prints "[Blinky Clyde Inky Pinky]"
```

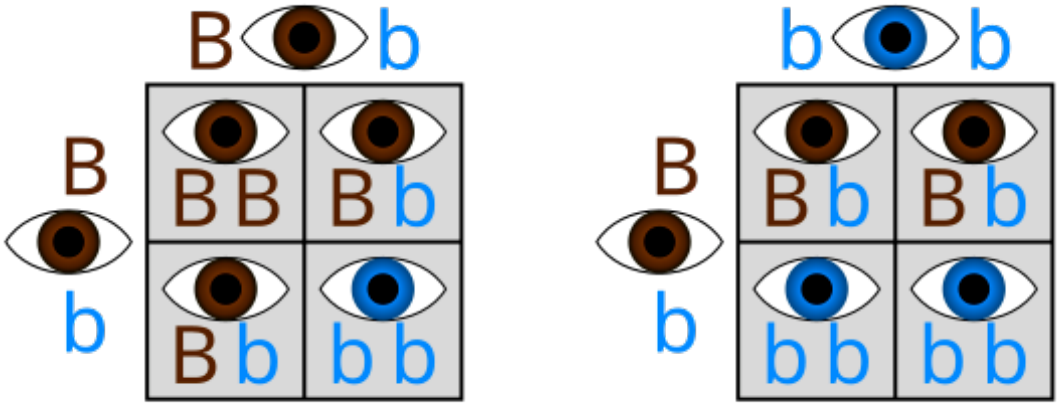
Programming: (40 points)

Write full programs in response to each prompt. You may use the Eclipse IDE to write and test your code.

Punnett Square (20 points)

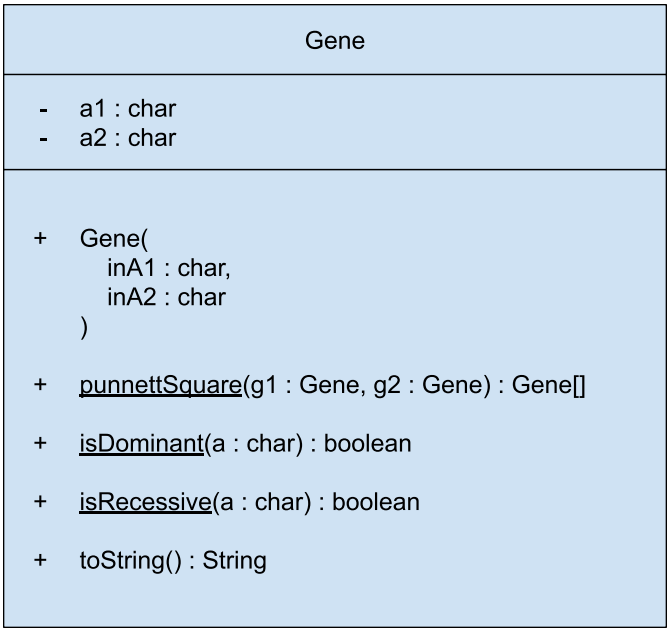
In Biology, genes are the method in which physical traits are passed down from parents to offspring. Each gene has two alleles representing either dominant traits or recessive traits. If an individual has at least one allele of the dominant trait, the phenotype (Actual physical appearance of the individual) will reflect the dominant trait. If instead the individual has two copies of the recessive gene, the phenotype will represent the recessive trait instead.

In the following images, the dominant allele for brown eye color is represented with a capital letter B, and the recessive allele for blue eye color is the lowercase b. A gene with the brown eye color allele (Uppercase "B") will always encode brown eyes. A gene with two copies of the blue eye allele (Lowercase "b") will encode blue eyes. A gene with one allele for brown eyes and one allele for blue eyes ("Bb" or "bB") will encode for brown eyes, as brown is dominant.



A tool called the "Punnette Square" is used to show how many possible offspring variations there could be for parents with specific genes. One parent's gene is placed on top of the square with each of the two alleles above an individual column, and the other parent's gene is placed on the side of the square with one allele next to each row. Then, four new genes are created by writing the alleles of the row and column together where they cross.

Write a class named Gene that holds two alleles as individual characters. Use the following UML diagram as a guide:



Above: UML Diagram of the Gene class.

The Gene class should have a constructor that takes in two characters as initial values for its alleles a1 and a2. There should be a non-static method toString() that returns a string of the two alleles. So, if Gene g has alleles a1 = 'B' and a2 = 'b', then g.toString() would return "Bb".
The Gene also has three static methods. isDominant and isRecessive take in a char and return a boolean. isDominant returns true if the allele char a is a dominant trait (Capital letter) and false otherwise. isRecessive returns true if the allele char a is a recessive trait (Lowercase letter) and false otherwise.

Last, the static punnetteSquare() method takes in two genes as input, and returns an array of four new genes which are the product of the punnett square operation on the two input genes. However, the genes must have compatible alleles (Same letters). I.E. If one gene has alleles "Aa" and another gene has alleles "Bb", the genes are not compatible because A and B encode for different traits. In the case of incompatible genes, return null.

NOTE: The order of the genes and the order of the alleles within the genes does not matter, as long as the correct four gene types are returned. So, the gene arrays [AA, Aa, Aa, aa] and [Aa, AA, aa, aA] are equivalent as they contain the exact same four gene combinations, just with a slightly different order of genes and alleles within the genes.

HINT 1: The String class has some useful functions for comparing Strings, and for converting Strings to uppercase or lowercase.

HINT 2: Remember that char data are integer values, and thus relational operators like >, <, >=, <=, and == can be used with chars just like with numbers. A short table of some helpful Ascii/Unicode values is below for reference.

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

Example:

```
public static void main(String[] args)
{
    Gene g1 = new Gene('A', 'a');
    Gene g2 = new Gene('A', 'a');
    Gene g3 = new Gene('B', 'b');
    Gene g4 = new Gene('b', 'b');

    Gene[] genes = Gene.punnettSquare(g1, g2);
    printGenes(genes);  // Prints: "AA Aa Aa aa "

    genes = Gene.punnettSquare(g1, g3);
    printGenes(genes);  // Prints: nothing, genes is null.

    genes = Gene.punnettSquare(g3, g4);
    printGenes(genes);  // Prints: "Bb Bb bb bb "
}

public static void printGenes(Gene[] genes)
{
    if (genes != null) {
        for (Gene g : genes) {
            System.out.print(g + " ");
        }
        System.out.println();
    }
}
```

Zoo (20 points)

Animaland is a zoo park that has many different animals. The zoo must keep track of various statistics related to its animals such as how many of each animal they have, that animal's diet, whether that animal is safe for visitors to pet, and the upkeep cost of that animal group.

Write a class named ZooAnimal that holds information about the Animaland animals. The class should have the following private fields:

- name: A String reference for a descriptive animal species name.
- population: An int holding the number of animals of that species.
- canPet: A boolean that is true if the animal is safe for visitors to pet, false otherwise.
- upkeep: A double with the cost of maintaining the animal group and its enclosure.

(10 points)
Write a constructor that accepts arguments for each field. Also write public getters and setters / accessors and mutators for each field.

I.E. For the field name, there should be methods getName() and setName(String). For field population, there should be methods getPopulation() and setPopulation(int). Etc.

(10 points)
Below is a table with information on some of the animals from Animaland's "Australian Outback" park section. Create three ZooAnimal instances using the info from the table below, then print out the information from your ZooAnimal instances. Use String.format, System.out.printf, and/or tabs ("\t") to help format your output in a clean and readable way. Your output should be printed in a format similar to the table below.

Species Name	Population	Safe to Pet?	Upkeep Cost
Koala	4	Yes	$1,100,000.00
Kangaroo	6	Yes	$750,000.00
Tasmanian Devil	2	NO!	$250,000.00

Fun Fact: Koalas, though very docile animals, are extremely expensive for zoos to keep due to their specific diet and specialized medical needs. Many zoos find the cost worth it, though, because their popularity offsets the high upkeep cost with significantly increased ticket sales.