Full Name: Utsav Shah

Student ID#: 1002158824

CSE 1325 OBJECT-ORIENTED PROGRAMMING

Exam #1 «---» 2 2 3 3 002 «---» Exam #1

Instructions

1. Students are allowed pencils, erasers, UTA Student ID, and beverage only. **A UTA Student ID is required to turn in the exam.**
2. All books, bags, backpacks, phones, **smart watches**, **ear buds**, and other electronics, etc. must be placed along the walls. **Silence all notifications.**
3. PRINT your name and student ID at the top of this page **and every additional pastel coding sheet**, and verify that you have all 10 pages.
4. **Read every question completely before you start to answer it.** If you have a question, please raise your hand. You may or may not get an answer, but it won't hurt to ask.
5. If you leave the room, you may not return.
6. You are required to SIGN and ABIDE BY the following Honor Pledge for each exam this semester.

Honor Pledge

On my honor, I pledge that I will not attempt to communicate with another student, view another student's work, or view any unauthorized notes or electronic devices during this exam. I understand that the professor and the CSE 1325 Course Curriculum Committee have zero tolerance for cheating of any kind, and that any violation of this pledge or the University honor code will result in an automatic grade of zero for the semester and referral to the Office of Student Conduct for scholastic dishonesty.

Student Signature: Utsav

Vocabulary

Write the word or phrase from the Words list below to the left of the definition that it best matches. Each word or phrase is used at most once, but some will not be used. {10 at 2 points each}

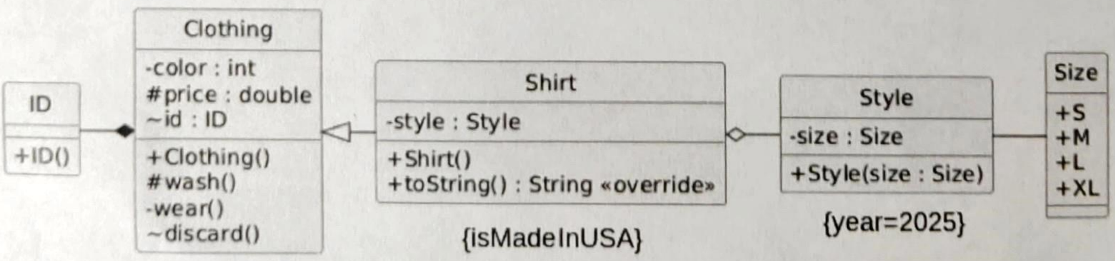
Vocabulary

Word	Definition
1 getter	A method that returns the value of a private variable
2 namespace	A named scope
3 definition	A declaration that also fully specifies the entity declared
4 interface	A reference type containing only method signatures, default methods, static methods, constants, and nested types
5 Abstract method	A method declared with no implementation
6 object	An instance of a class containing a set of encapsulated data and associated methods
7 destructor	A special class member that cleans up when an object is deleted
8 variable	A block of memory associated with a symbolic name that contains a primitive data value or the address of an object instance
9 assertion	An expression that, if false, indicates a program error
10 abstract class	A class that cannot be instantiated

Word List

Abstract Class	Abstract Method	Abstraction	Algorithm	Assertion
Class	Constructor	Data Validation	Declaration	Definition —
Destructor	Encapsulation	Enumerated Type	Exception	Field
Garbage Collector	Getter	Inheritance	Interface	Method
Multiple Inheritance	Namespace	Object	Object-Oriented Programming	Operator
Override	Package	Primitive type	Setter	Subclass
Superclass	UML	Validation Rules	Variable	Version Control

Figure 1: Will be referenced by questions in Multiple Choice section (next)



20

Multiple Choice

Read the full question and every possible answer. Choose the one best answer for each question and write the corresponding letter in the blank next to the number. The image below clarifies the meaning of open and closed arrows and diamonds. Note that "e.g." means "for example". {15 at 2 points each}

1. D **A Java subclass may inherit from**

   A. Either a superclass or an interface

   B. Both one superclass and one interface

   C. Any number of superclasses and interfaces

   D. One superclass and any number of interfaces

2. D **Class `Book` is an abstract class with a default constructor. Which of the following will create an instance of class `Book`?**

   A. `Book book = new Book();`

   B. `Book book();`

   C. `Book book;`

   D. An abstract class cannot be instanced

3. D **Refer to Figure 1 on page 2. The relationship between `Clothing` and `ID` is**

   A. Inheritance

   [✓ B. Composition]

   C. Association Class

   D. Dependency

4. C **Documentation for Java packages is usually written using**

   A. Microsoft Word

   B. GitHub Markdown

   C. Javadoc

   D. README.txt files in the same directory

5. D **A final method**

   A. cannot be edited

   B. is called when an object is deleted

   C. cannot modify fields

   D. cannot be overridden

8

6. B Which statement is TRUE about `String` in Java?

A. In Java, `String` is another name for `char*`

B. String is "immutable" and thus cannot be changed once constructed

C. A specific `char` within a `String` may be accessed with subscripts like `s[3]`

D. `String` is a primitive type in Java

7. A Which is TRUE about Java exceptions?

A. An uncaught exception will cause the program to abort

B. Only the main method can catch an exception

C. An exception, once caught, cannot be rethrown

D. Any variable, even a primitive, may be thrown in Java (but throw exceptions please!)

8. B Refer to Figure 1 on page 2. The `id` field for class `Clothing` is

A. Public

B. Package-private

C. Protected

D. Private

9. B The class version of Java's array is the

A. `ArrayClass`

B. `ArrayList`

C. `Array`

D. `ListClass`

10. A A successful regression test should print

A. Nothing

B. "Pass"

C. "0"

D. "1"

10

11. B **Operator == compares**

[✓ A. the memory address of two objects, but the values of two primitives]

B. The memory address of two objects or primitives

C. the value of two objects, but the memory address of two primitives

D. The value of two objects or primitives

12. B **Refer to Figure 1 on page 2. The "{isMadeInUSA}" under class `Shirt` is a**

A. Stereotype

B. Constraint

C. Comment

D. Tag

13. A **To control how an object is converted into a String, for example as a parameter to `System.println`, we would**

A. Write a `format` function that returns the string representation

[✓ B. Override the `toString()` method]

C. Use Javadoc

D. Specify an `sprintf` format string as a template

14. C **The difference between `System.out` and `System.err` is**

A. `System.out` doesn't check for errors, while `System.err` does

B. `System.out` uses `println`, while `System.err` uses `printf`

C. `System.out` streams out data, while `System.err` streams out error messages

D. They both do exactly the same thing

15. A **Which of the following ends the program and reports `42` to the operating system?**

A. `System.exit(42);`

B. `System.return(42);`

C. `return 42;`

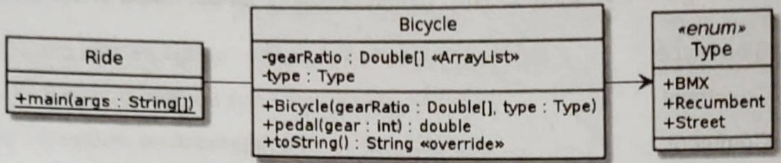
D. `return(42);`

6

Free Response

Write the solutions to the Free Response questions in the provided space. Additional coding sheets are available on request. **Write your name and student ID on EVERY additional coding sheet you use that is not already stapled to this exam.**

Each question stands alone, with instructions on what to code from the class diagram. If you have trouble answering a question, simply move to the next question and come back later. **Assume all needed imports - don't code them.**


Ride
+main(args : String[])
Bicycle
-gearRatio : Double[] «ArrayList»
-type : Type
+Bicycle(gearRatio : Double[], type : Type)
+pedal(gear : int) : double
+toString() : String «override»
«enum»
Type
+BMX
+Recumbent
+Street

1. {code an enum, 3 points} In file Type.java, code ONLY enum `Type` from the class diagram above so that it would be visible in all packages in the application.

```
public enum Type {
        BMX, Recumbent, Street
    }
```

3

2. {code a class} For this question, consider the `Bicycle` class in the class diagram above.

a. {5 points} In file Bicycle.java, write the class (such that it is visible across all packages) and field declaration but omit the methods (they are covered below).

```
public class Bicycle {
        private ArrayList<Double> gearRatio;  *
        private Type type;
```

5

b. {6 points} In file Bicycle.java, write the `Bicycle` class constructor. Note that `gearRatio` is an `ArrayList`. If `gearRatio` is `null` OR empty, throw an `IllegalArgumentException` with the message "Bad gearbox". Otherwise, assign each field to its corresponding parameter.

*. check type -0.5.

5.5

```
public Bicycle (Type type, Double[]gearRatio){
    if (gearRatio == null || gearRatio.isEmpty()) {
        throw new IllegalArgumentException("Bad
                                      gearbox");
    }
    this.type = type;
    this.gearRatio = gearRatio;
}
```

c. {5 points} In file Bicycle.java, write the `pedal` method. If the parameter `gear` (an `int`) is a valid index for ArrayList `gearRatio`, return the ratio value (a `Double`) at index `gear` in `gearRatio`, otherwise return -2.0.

```
public double pedal ( int gear) {
    for (int i = 0; i < gearRatio.size(). ~~size()~~ ; ++i) {
                                  → check if gear idx is within
        if ( i.equals(gear)) {      bounds of gearRatio   -1.
3.
            return gearRatio[gear];
                                Not an array
        }                       gearRatio.get(gear)  - 1.
    }
    return -2.0;
}
```

d. {7 points} In file Bicycle.java, override the `toString()` method such that a compiler error will be generated if the superclass has no matching method.

Using a `StringBuilder` object, return the `type` of bicycle, number of gears, and the `gearRatio` values separated by commas (for ANY number of gears).

For example, a BMX bike with gear ratios of 1.1, 2.5, 4.2, and 9.9 would return

```
BMX 4-speed with ratios 1.1, 2.5, 4.2, 9.9
```

```
                                     * sb has an append method.
@Override
public String toString() {
    StringBuilder result = new StringBuilder();
    result = myBicycle.type;   .append(type).  -0.5.
3.5.
    result.*add(" ");
    result.add(gear);  , append gearRatio.size() . -1.
    result.add("-speed with ratios ");
    for (int i = 0; i < gear; i++){
        result.add(gearRatio[i]);    }  - 1.5 for each loop
    }                                   over gearRatio
                                        to append ratio.
    return result;
                 .toString()  -0.5.
  }
}              return type's a string not a sb.
```

3. {code a main method, 8 points} For this question, code the `Ride` class in the class diagram above. Do NOT write imports; assume you have what you need.

In the main method, instance a `Bicycle` of type `Street` with gear ratios 1.7, 4.2, and 5.7.

Print your `Bicycle` instance.

Then call method `pedal` on it, selecting the middle gear (gear 1), and print to the standard out stream "Pedaling at ratio " and the result of method `pedal`.

If an `IllegalArgumentException` occurs at any point during the above, print its message to the standard error stream and exit the program with error code -3.

If correctly written, your program should have the following output:

```
Street 3-speed with ratios 1.7, 4.2, 5.7
Pedaling at ratio 4.2
```

(-1 1/2) instance a ArrayList & local ~~dob~~ ratios (e)

```
public static void main(String[] args) {
try{ Bicycle myBicycle = new Bicycle(~~Street~~, [1.7, 4.2, 5.7]);
   System.out.printf("\n %s ", myBicycle.type);
    System.out.printf ("%n-speed with ratios ", gear);
   for(int i=0 ; i<3 ; ++i) {
     System.out.print (myBicycle.gearRatio [i]);
   }

   ~~try {~~
   double a = pedal(1);
   System.out.printf("\n Pedaling at ratio %d\n",
                         a);
   }
   catch (IllegalArgumentException e){
   System.err.println (e.message);
   System.exit(-3);
   }
}
```

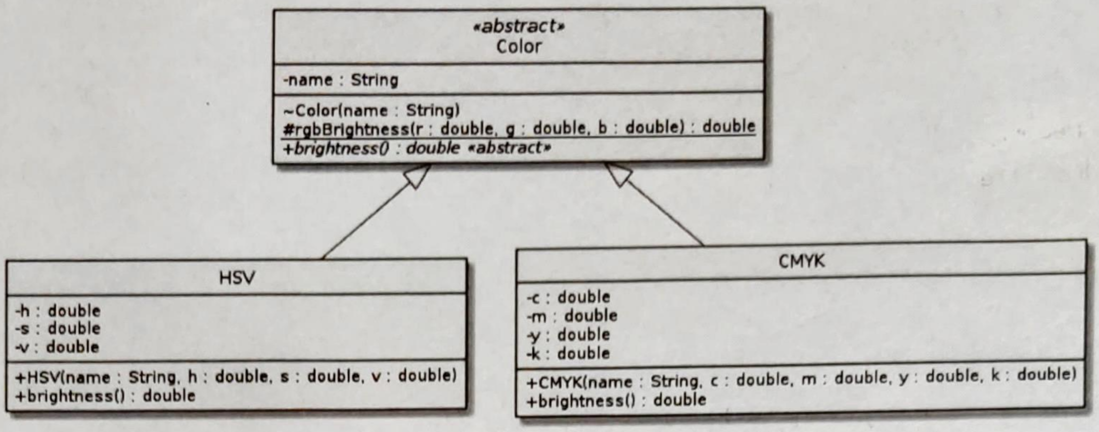
Type.Street (-1/2)

(-1/2)

✗ middle gear: (-1/2)

5

4. {inheritance} Consider the class diagram below.


«abstract»
Color
-name : String
~Color(name : String)
#rgbBrightness(r : double, g : double, b : double) : double
+brightness() : double «abstract»
HSV
-h : double
-s : double
-v : double
+HSV(name : String, h : double, s : double, v : double)
+brightness() : double
CMYK
-c : double
-m : double
-y : double
-k : double
+CMYK(name : String, c : double, m : double, y : double, k : double)
+brightness() : double

a. {7 points} In file Color.java, write all of abstract class `Color`. The package-private constructor simply assigns its parameter to the field. Static method `rgbBrightness` returns the double value `0.2126*r + 0.7152*g + 0.0722*b`. Also code `brightness`.

```
abstract class Color {
    private String name;
                   blank -1/2
4½  package-private Color (String name) {
        this.name = name;
    }
    protected -1/2
    public static double rgbBrightness(double r, double g,
        double b) {
        double result = 0.2126*r + 0.7152*g + 0.0722*b;
        return result;
    }
 -1/2
public  abstract double brightness( ){ ; -1

    }
}
```

b. {9 points} In file HSV.java, write all of the subclass `HSV`. The constructor constructs all inherited and local fields from the parameters per Java requirements. The `brightness` method (for which the compiler MUST verify is being overridden) must return the result of `Color`'s static method `rgbBrightness(v/0.0709, v/0.2384, v/0.0241).`

```
public class HSV extends Color {
        private double h;
        private double s;
        private double v;
                  String name,
     public HSV(double h, double s, double v) {
          this.h = h;   super(name); -1
          this.s = s;
          this.v = v;
     }
     @Override
      public double brightness () {
   return rgbBrightness(v/0.0709, v/0.2384, v/0.0241) }
}
```

8

Bonus

BONUS 1: {+4 points} `String s` has size of at least 3 chars. Demonstrate up to 4 *different* Java algorithms for determining if the first 3 characters of `s` are "CSE".

0

BONUS 2: {+3 points} In no more than TWO concise sentences, explain the difference between a permissive and a share-alike software license and give a major example license of each type.

0