Full Name: _____

Student ID#: _____

CSE 1325 OBJECT-ORIENTED PROGRAMMING

PRACTICE #1 Exam #2 «---» 1 1 001 «---» Exam #2 PRACTICE #1

Instructions

1. Students are allowed pencils, erasers, and beverage only.
2. All books, bags, backpacks, phones, **smart watches**, 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 coding sheet**, and verify that you have all 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: _____

WARNING: Questions are on the BACK of this page!

Vocabulary

Write the word or phrase from the Word 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}

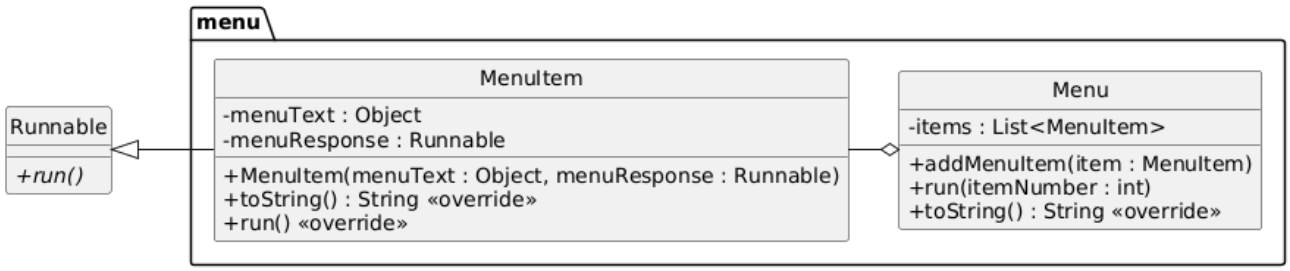
Vocabulary

Word	Definition
1	Reuse and extension of fields and method implementations from another class
2	Read-only memory for machine instructions
3	A style of programming focused on the use of classes and class hierarchies
4	A standard library abstraction for objects referring to elements of a container
5	A self-contained execution environment including its own memory space
6	A class member variable
7	A template encapsulating data and code that manipulates it
8	Memory for declarations outside of any class or function scope
9	Scratch memory for a thread of execution
10	A grouping of related types providing access protection and namespace management

Word List

Abstraction	Algorithm	Attribute	Class	Class Hierarchy
Class Library	Code	Concurrency	Constructor	Encapsulation
Garbage Collector	Generic	Generic Programming	Global	Heap
Inheritance	Interface	Invariant	Iterator	Method
Mutex	Object	Object-Oriented Programming (OOP)	Package	Polymorphism
Process	Reentrant	Stack	Synchronized	Thread

NOTE: This class diagram is referenced by question 1.i.



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. {15 at 2 points each}

1. ____ **To declare a Java method with two generic types, write**

    A. `public <K, V> V getElement(K key)`

    B. `public <K> <V> V getElement(K key)`

    C. `public V getElement <K><V> (K key)`

    D. `public <V> V getElement(<K> K key)`

2. ____ **Thread interference may be fixed by using**

    A. A synchronized method

    B. A synchronized thread object

    C. A getter and a setter

    D. A synchronized destructor

3. ____ **To use a class as the key for a** `TreeMap`**, it must implement interface**

    A. `AutoCloseable`

    B. `Runnable`

    C. `Iterable`

    D. `Comparable`

4. ____ **To determine if** `Iterator it` **could return the next element, write**

    A. `if(it != null)`

    B. `if(it.hasNext())`

    C. `if(it.remaining > 0)`

    D. `if(it.size() > 0)`

5. ____ **A** `LinkedList` **is generally preferred instead of an** `ArrayList` **when**

    A. More gets will be performed than inserts and deletes

    B. The number of data elements to be stored is not known in advance

    C. More inserts and deletes will be performed than gets

    D. Memory needs to be conserved

6. ____ **To append to rather than overwrite an output file in Java,**

    A. Open the output file with `FileWriter` and then call its `append()` method

    B. Open the output file with `FileAppender` instead of `FileWriter`

    C. Open the output file with `FileWriter` and then call its `seek(size()-1)` method

    D. Set the second parameter of the `FileWriter` constructor to true

7. ____ **A Java** `ListIterator` **is like a C pointer except that it cannot**

    A. advance to the next element (`it.next()`)

    B. return to the previous element (`it.previous()`)

    C. overwrite the current element (`it.set(newValue)`)

    D. do pointer math (`it.next(+3)`)

8. ____ **To invoke the default superclass constructor from a subclass constructor, write**

    A. `chain();`

    B. `super();`

    C. `this();`

    D. the name of the superclass, such as `Foo();`

9. ____ **Which code will NOT return an element of** `HashSet<Integer> s`**?**

    A. `return s.iterator().next();`

    B. `return s.get(0);`

    C. `for(int i : s) return i;`

    D. `return (Integer) s.toArray()[0];`

10. ____ **Which Java method in superclass Animal could be polymorphically called in its subclasses?**

    A. `private void walk();`

    B. `final void walk();`

    C. `polymorphic void walk();`

    D. `public void walk();`

11. ____ **Given** `BufferedReader br`**, when** `br.readLine()` **is called on a stream that has reached end of file,**

    A. It will return `null`

    B. It will return an empty String

    C. It will throw an `EndOfFileException`

    D. It will return -1

12. ____ **Method** `public void update(List<?> list)` **is a**

    A. Generic method with a `List` parameter supplied when the method is called

    B. Generic method with a `List` parameter containing any object type

    C. Generic regular express method with a `List` parameter accepting any single-character type

    D. Compiler error, as ? is not a valid type in Java

13. ____ **To pause a Java program for at least 6 seconds, use**

    A. `for (int i=0; i<6000000; ++i) { }`

    B. `Time go=now()+6; while(now() < go);`

    C. `Thread.sleep(6000);`

    D. `System.pause(6);`

14. ____ **The parameter to the Thread constructor we used in class is an object that implements**

    A. `Iterable`

    B. `Comparable`

    C. `AutoCloseable`

    D. `Runnable`

15. ____ **Which of these types is defined as generic (all are real library classes)?**

    A. TreeSet

    B. FileWriter

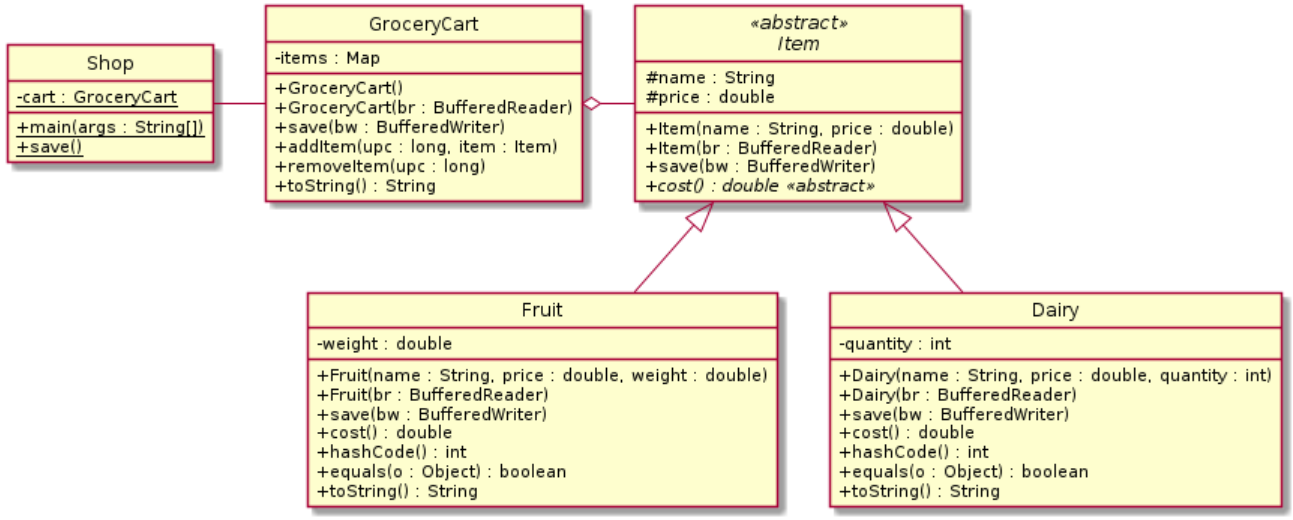
    C. BufferedReader

    D. Double

Free Response

Provide clear, concise answers to each question. Each question may implement only a portion of a larger Java application. Each question, however, is *completely independent* of the other questions, and is intended to test your understanding of one aspect of Java programming. **Write only the code that is requested.** You will NOT write entire, large applications! **Additional paper is available on request.**

1. {polymorphism, collections, iterators, file I/O} Consider the following class diagram representing grocery shopping for food. The `GroceryCart` class stores and removes `Item` objects in `Map items` using the `addItem` and `removeItem` methods, respectively, using `long upc` as the key. The diagram shows two subclasses of `Item`, `Fruit` (measured by weight) and `Dairy` (measured by quantity). All classes can save and restore themselves. You will NOT write much of this application.



a. {2 points} Assume a line of code: `Item item = new Fruit ( ... );` (do NOT write parameters in place of `...`).

- If this code won't compile, write "Won't compile."
- If this code will compile but will throw an exception, write the exception that will be thrown.
- If this code will compile and NOT throw an exception, write just one line of code to *polymorphically* get the cost of the `Fruit` object from `Item` variable `item` using overridden method `cost()`.

b. {4 points} Write `Fruit.equals(Object o)` . All 3 fields (including inherited fields) are significant.

c. {3 points} Write `Fruit.hashCode()`. All 3 fields (including inherited fields) are significant.

d. {3 points} Write the `addItem` method in `GroceryCart` that adds its `upc` and `item` parameters to the `items` Map.

e. {3 points} Write the `removeItem` method in `GroceryCart` that removes the pair from the `items` Map where the key matches the `upc` parameter.

f. {3 points} Write the `save` method for `Fruit` class. Write all inherited fields to `bw` without referencing them, then write the field defined in `Fruit` and do NOT catch the *checked* exception `IOException`.

g. {3 points} Write the `Fruit` constructor with the `BufferedReader` parameter. Construct a new `Fruit` by restoring the inherited fields first, then the field defined in `Fruit` and do NOT catch the *checked* exception `IOException`.

h. {6 points} Write `GroceryCart.toString()`, which formats the UPC codes (the key in Map `items`) and Item objects (the value in Map `items`) into a receipt that looks similar to this. (I recommend "%15d" for the UPC codes. The Fruit and Dairy toString() methods, which you should NOT write, format the rest.)

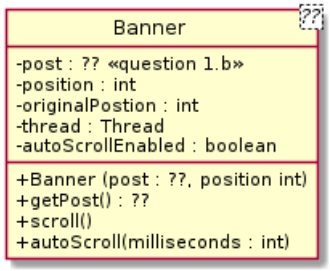
```
           Receipt

       4011            Bananas (2.1 # @  0.69  per)  $1.46
21000001927   8 oz Medium Cheddar ( 2 @  3.29 each)  $6.58
```

- Instance a `StringBuilder` object with the "Receipt" header.
- Obtain the set of keys from Map `items` .
- Obtain an iterator from the set of keys. (You will receive no credit unless you use an iterator.)
- While additional keys remain, obtain the next key and use it to obtain the associated item, then append both to the `StringBuilder` object.
- Return the String version of the `StringBuilder` object.

i. {3 points} Write the main method in class `Shop`. Instance a new `GroceryCart` referenced by field `cart`. Using the `Menu` and `MenuItem` classes discussed in lecture (see class diagram on page 2), instance `Menu` and add just one `MenuItem` instance to it: "Save" that calls method `save()`. Then use your `Menu` instance to dispatch the selection of "Save" as if by the user.

j. {3 points} In class `Shop`, write ONLY the `save()` method. Use a try-with-resources to open file "cart.txt" and tell the `cart` instance to save itself. If an `IOException` is thrown, print it to the console's error stream.

2. {generics, threads} Consider the class diagram.



Generic class Banner scrolls the toString representation of field `post` across the terminal. The constructor accepts an object of its one generic type which is stored in the `post` field. The `scroll()` method prints the number of spaces defined by `position` followed by `post` on first call, with one less space on each subsequent call, giving the effect of the text scrolling right to left on the terminal. Method `autoScroll(int milliseconds)` scrolls the object in a separate thread, one character every `milliseconds` until `autoScrollEnabled` becomes false, while the main thread continues executing.

a. {2 points} Write the one-line declaration for generic class Banner . You may use any generic variable you please and you may declare you are implementing any interfaces you believe you need.

b. {2 points} Write the one-line definition of private field `post` of the generic type.

c. {3 points} Write the constructor for class Banner that accepts a `post` parameter of the generic type and a `position` parameter of type int. Assign the parameters to fields of the same name. Also assign `position` to field `originalPosition`.

d. {2 points} Write the `getPost()` getter, which simply returns field `post` as its native type (that is, you may NOT just return type Object).

e. {2 points} Write Java code demonstrating how you would ensure that only one thread could execute method `scroll()` at a time. You may write the method declaration (do NOT write the entire method!) OR demonstrate how to call the method from within a thread while avoiding thread interference.

f. {6 points} Write method `autoScroll(int milliseconds)`.

- First, if field `thread` is not null, set `autoScrollEnabled` to false and wait for the Thread referenced by `thread` to exit.
- Next, if parameter `milliseconds` is less than 0, return. (Thus, calling `autoScroll` with a negative parameter just terminates autoscrolling.)
- Finally, set `autoScrollEnabled` to true and run a new thread. The thread body loops while `autoScrollEnabled` is true, calling method `scroll()` and then pausing for `milliseconds` milliseconds each iteration. You are permitted to write a separate method for the thread body, or you may use an anonymous class or a lambda for the thread body as you please. **Do** handle *checked* exception `InterruptedException` here in any (valid) way you please.

Bonus

Bonus {+4 points} Given `class Animal`, `class Dog extends Animal`, `Animal animal`, and `Dog dog`, give an example of the following:

- Upcast (if this might fail, set the variable to null if it does):

- Downcast (if this might fail, set the variable to null if it does):