Due at end of session

This exam is: **closed-book**, **NO electronic devices allowed**, and **closed-notes**. The exception is the “sage page” of the designated size on which you may have notes to consult during the exam.

Be sure you: **Provide legible answers in designated areas** (credit will not be given for work that is difficult to read or not where expected), Ensure you clearly fill in a single circle on multiple choice questions, Use indentation of your code to show its structure (but don’t dwell on exact punctuation/syntax), Leave the exam stapled together in its original order, Do **NOT** attach any other pages to the exam. You are welcome to use the blank space on the exam for any scratch work.

If you need to leave the room for any reason prior to turning in your exam, you must leave your exam and any electronic devices with a proctor.

You must complete all the identifying information below correctly. Failure to do so is grounds for a zero on this exam:

1. **Name** (print clearly): __________________________________________________________

2. **Student ID** (print clearly; 1 digit per underline): ___ ___ ___ ___ ___ ___ ___

3. Which time do you typically attend studio (fill one)
   - [ ] 11:30—1:00  
   - [x] 1:00—2:30
   - [ ] 2:30—4:00
   - [ ] 4:00—5:30

4. Which Urbauer lab do you typically sit in (best guess; fill one)
   - [ ] 214
   - [x] 216
   - [ ] 218
   - [ ] 222

5. You must sign the pledge below for your exam to count. The penalty for cheating will be decided during academic integrity review, but the instructors will recommend an F in this course as the minimum penalty.

   I have read the instructions on this page and I will neither give nor receive any unauthorized aid on this exam.

---

(Sign above)

⇒ Do not proceed until told to do so! ⇐

⇒ Initial the top right corner of each page before starting ⇐
1. (6 points) Indicate whether an example of each Java data type is an object or not:

(1) String  ○ Object  ○ Not an Object
(2) int  ○ Object  ○ Not an Object
(3) Color  ○ Object  ○ Not an Object
(4) double  ○ Object  ○ Not an Object
(5) HashMap  ○ Object  ○ Not an Object
(6) ArgsProcessor  ○ Object  ○ Not an Object

2. (8 points) True or False:

(1) If a method passes JUnit tests for that method, the method will always work with all inputs:
   ○ True  ○ False

(2) Methods must be declared as public:
   ○ True  ○ False

(3) A class may only have one constructor method:
   ○ True  ○ False

(4) Instance variables (fields) that are declared final can be assigned in any method:
   ○ True  ○ False

(5) The static methods of a class can access instance variables:
   ○ True  ○ False

(6) Given two String objects x and y, if x == y is true then x.equals(y) must also be true.
   ○ True  ○ False

(7) Given two String objects x and y, if x.equals(y) is true then x == y must also be true.
   ○ True  ○ False

(8) Given two String objects x and y, if x.equals(y) is true then y.equals(x) must also be true.
   ○ True  ○ False

3. (1 point) Objects are allocated in the:
   ○ Garbage Collector  ○ Heap  ○ Objectifier  ○ Stack

4. (1 point) In Java, a valid hashCode() method should return a(n):
   ○ double  ○ HashMap  ○ int  ○ String

5. (1 point) In Java, signatures for methods that may be implemented by many classes are provided by a(n):
   ○ class template  ○ implements  ○ interface  ○ method manifest
6. (8 points) Consider the three major “collection” ADTs covered: List, Set, and Map. Indicate which is the best answer or the best fit for the described problem:

(1) Which is always ordered:
   - List  ○ Map  ○ Set

(2) Which has two parts: keys and associated values:
   - List  ○ Map  ○ Set

(3) The computer science department would like to know all the distinct students enrolled in CSE courses. That is, we’d like to build a collection from the rolls of students in all classes, but a single student should only be in the collection once. Which is best:
   - List  ○ Map  ○ Set

(4) An Android App needs to match a city name to its zip code. Which is best:
   - List  ○ Map  ○ Set

(5) A program needs to keep track of the students in a waiting list. Which is best:
   - List  ○ Map  ○ Set

(6) One of the three previous parts (parts 3, 4, and 5) will use an ADT that has both a key and a value.
   i. What concept will be used for the key:
      - student  ○ city name  ○ zip code  ○ homework assignments  ○ due date
   
   ii. Fill in the blank. What will the data type of the key be: ______________________

   iii. What concept will be used for the value:
      - student  ○ city name  ○ zip code  ○ homework assignments  ○ due date

7. (3 points) OOP Terminology

   (1) Class methods that change a property of an object are generically referred to as:
      - Constructors  ○ Getters  ○ Setters  ○ Transmogrifiers

   (2) Class methods that are called when new is used are referred to as:
      - Constructors  ○ Getters  ○ Setters  ○ Transmogrifiers

   (3) Class methods that return a property of an object are generically referred to as:
      - Constructors  ○ Getters  ○ Setters  ○ Transmogrifiers
8. Given the following class, show everything that will be printed to the console by each example piece of code shown. Be as accurate as possible, including spacing and line usage. Output should start on the first dotted line on the right side of the paper (the leftmost dot corresponds to the left side of the console window).

Correct answers should use each and every dotted line.

```java
public class VersionNumber {
    private int major;
    private int minor;

    public VersionNumber(int major) {
        System.out.println('C');
        this.major = major;
        this.minor = 1;
    }

    @Override
    public String toString() {
        String version = 'v' + major + '.' + minor;
        System.out.println('S: ' + version);
        return version;
    }

    boolean isEquivalent(VersionNumber other) {
        String instance = this.toString();
        return instance.equals(other.toString());
    }

    public void minorBump() {
        System.out.println('L: ' + this);
        minor = minor + 1;
    }

    public void majorBump() {
        System.out.println('B');
        major = major + 1;
        minor = 0;
    }
}
```

(1) (1 point)
```
VersionNumber v2 = new VersionNumber(2); .............................................
```
This continues the problem on the previous page.
Correct answers should use each and every dotted line.

(2) (5 points)
VersionNumber v3 = new VersionNumber(3);
v3.majorBump();
System.out.println(v3); ...........................................................

(3) (5 points)
VersionNumber v4 = new VersionNumber(4);
VersionNumber v4b = new VersionNumber(4);
v4.majorBump();
System.out.println(v4.isEquivalent(v4b)); ...........................................................

(4) (5 points)
VersionNumber v5 = new VersionNumber(5);
v5.minorBump();
System.out.println(v5); ...........................................................
9. (14 points) Complete the code below to: a) create a mirror of a `Point3d`, b) to determine if two points are at the same location, and c) determine the distance from another point.

```java
public class Point3d {
    private final double x;
    private final double y;
    private final double z;
    public Point3d(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }
    public double getX() {
        return x;
    }
    public double getY() {
        return y;
    }
    public double getZ() {
        return z;
    }
}
```

(1) (4 points) Complete the method to create a mirror of a `Point3d`:

```java
/**
 * Creates a point which mirrors this one.
 * For example, an instance with x=1, y=-2, z=3 would return a
 * Point3d whose x=-1, y=2, z=-3.
 * @return a mirrored version of this point
 */
public Point3d createMirror() {
    // Add code here
    return new Point3d(-x, -y, -z);
}
```
This continues the problem on the previous page.

(2) (3 points) Complete the method to determine if two points are at the same location:

```java
@override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if ((obj == null) || (getClass() != obj.getClass()))
        return false;
    Point3d other = (Point3d) obj;
```

(3) (7 points) Complete the method to determine the distance from another point. Fill in the lines for parameters (it should only have one) and return type as well as the body of the method:

```java
public static ___________________ distanceTo(____________________________) {
```

Consider the following interface for the next two problems:

```java
public interface ProvidesVolume {
    double getVolume();
}
```

And that the `Math` class provides a value for π:

```java
public class Math {
    public static final double PI = 3.14159265358979323846;
    ...
}
```

10. (5 points) Complete the `Sphere` class. You should fill what is necessary to implement the `ProvidesVolume` interface. The volume of a sphere is:

\[ V = \frac{4\pi r^3}{3} \]

```java
public class Sphere implements ProvidesVolume {
    private final Point3d center;
    private final double radius;

    public Sphere(Point3d center, double radius) {
        this.center = center;
        this.radius = radius;
    }

    // Complete what is necessary to implement ProvidesVolume:

    ............................................................
    ............................................................
    ............................................................
    ............................................................
    ............................................................
    ............................................................
}
```
11. (8 points) Complete the `Box3d` class (it’s an axis aligned box). You should fill in the methods `getWidth()` for the x dimension, `getHeight()` for the y dimension, and `getDepth()` for the z dimension, as well as completing what is necessary to implement the `ProvidesVolume` interface. The volume of a box is: $V = \text{width} \times \text{height} \times \text{depth}$

```java
public class Box3d implements ProvidesVolume {
    private final Point3d llf;
    private final Point3d rub;
    public Box3d(Point3d leftLowerFront, Point3d rightUpperBack) {
        this.llf = leftLowerFront;
        this.rub = rightUpperBack;
    }
    public double getWidth() {
        // Complete what is necessary to implement ProvidesVolume
    }
    public double getHeight() {
        // Complete what is necessary to implement ProvidesVolume
    }
    public double getDepth() {
        // Complete what is necessary to implement ProvidesVolume
    }
    // Complete what is necessary to implement ProvidesVolume
}
```
Below you will shortened versions of the JavaDocs for methods found in `interface Map<K,V>`. Use this as a reference for the two problems that follow.

**boolean containsKey(K key)**
Returns true if this map contains a mapping for the specified key. More formally, returns true if and only if this map contains a mapping for a key k such that (key==null ? k==null : key.equals(k)). (There can be at most one such mapping.)

**boolean containsValue(V value)**
Returns true if this map maps one or more keys to the specified value. More formally, returns true if and only if this map contains at least one mapping to a value v such that (value==null ? v==null : value.equals(v)).

**Set<Map.Entry<K,V>> entrySet()**
Returns a Set view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa.

**V get(K key)**
Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

**V getOrDefault(K key, V defaultValue)**
Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key.

**Set<K> keySet()**
Returns a Set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa.

**V put(K key, V value)**
Associates the specified value with the specified key in this map (optional operation). If the map previously contained a mapping for the key, the old value is replaced by the specified value. (A map m is said to contain a mapping for a key k if and only if m.containsKey(k) would return true.)

**int size()**
Returns the number of key-value mappings in this map.

**Collection<V> values()**
Returns a Collection view of the values contained in this map. The collection is backed by the map, so changes to the map are reflected in the collection, and vice-versa.
12. (9 points) The code below will have two parts. We have provided the first part which: a) prompts the user via ArgsProcessor for a number of (key, value) pairs and then b) loops that many times, again prompting the user, and associates each key with its value.

Your task is to write the second part, which should: a) print a heading and b) print all of the (key, value) pairs in the map. For example, if the user entered:

```
3 a 1.0 b 2.0 c 3.0
```

then the output in the Console should look like something reasonably approximating:

```
console
printing 3 (key, value) pairs:
a -> 1.0
b -> 2.0
c -> 3.0
```

NOTE: Do not concern yourself about the ordering of the (key, value) pairs, but you should use the format shown above.

```java
Map<String, Double> map = new HashMap<String, Double>();
ArgsProcessor ap = new ArgsProcessor(args);
int n = ap.nextInt("How many key value pairs?");
for (int i = 0; i < n; ++i) {
    String key = ap.nextString("key?");
    Double value = ap.nextDouble("value?");
    map.put(key, value);
}
```
13. (6 points) The code below will also have two parts. The first part, which fills a map with letter grades mapped to the points each is worth and then prompts the user via ArgsProcessor for a letter grade to lookup, is completed for you.

Your task is to write the second part, which must lookup the user specified string in letterGrade. If letterGrade is associated with a value in the map, then print that value. Otherwise, print -1.0.

```java
Map<String, Double> map = new HashMap<String, Double>();
map.put("A+", 4.0);
map.put("A", 4.0);
map.put("A-", 3.7);
map.put("B+", 3.3);
map.put("B", 3.0);
map.put("B-", 2.7);
map.put("C+", 2.3);
map.put("C", 2.0);
map.put("C-", 1.7);
map.put("D+", 1.3);
map.put("D", 1.0);
map.put("D-", 0.7);
map.put("F", 0.0);

ArgsProcessor ap = new ArgsProcessor(args);
String letterGrade = ap.nextString("Letter grade to look up?");
```