AP Computer Science \_\_\_\_\_\_\_ Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Quiz: Classes, Inheritance 25 points Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Period \_\_\_\_\_\_

and 1D Arrays

AP Quick Reference Sheet Allowed

**Multiple Choice (1 point each): Choose the best answer and circle it on your quiz.**

1. Assuming all variables are declared correctly, which of the following swaps two elements in an array of integers called numbers at the index of x and y?
2. numbers[x] = numbers[y];

numbers[y] = numbers[x];

1. int temp = numbers[x];

numbers[y] = numbers[x];

numbers[y] = temp;

1. int temp = numbers[x];

numbers[x] = numbers[y];

numbers[y] = temp;

1. I only
2. II only
3. III only
4. II and III
5. I, II, and III
6. What does the following code do? Assume list is an array of int values, temp is some previously initialized int value, and c is an int initialized to 0.

for (j = 0; j < list.length; j++){

if(list[j] < temp) {

c++;

}

}

1. It finds the smallest value and stores it in temp
2. It finds the largest value and stores it in temp
3. It counts the number of elements equal to the smallest value in list
4. It counts the number of elements in list that are less than temp
5. It sorts the values in list to be in ascending order

**For questions 3 – 4, assume an int array, candy, stores the number of candy bars sold by a group of children where candy[j] is the number of candy bars sold by child j. Assume there are 12 children in all.**

3. Which of the following code segments could be used to compute the total number of bars sold by the children?

1. for (int j = 0; j < 12; j++)

sum += candy[j];

1. for (int j = 0; j < 12; j++)

candy[j] = sum;

1. for (int j = 0; j < 12; j++)

sum = candy[j];

1. for (int j = 0; j < 12; j++)

sum += [j];

1. for (int j = 0; j < 12; j++)

[j] += sum;

4. What does the following method do?

public int question5()

{

int value1 = 0;

int value2 = 0;

for (int j = 0; j < 12; j++){

if(candy[j] > value1)

{

value1 = candy[j];

value2 = j;

}

}

return value2;

}

1. It returns the total number of candy bars sold.
2. It returns the total number of children who sold 0 candy bars.
3. It returns the total number of children who sold more than 0 candy bars.
4. It returns the number of candy bars sold by the child who sold the most candy bars.
5. It returns the index of the child who sold the most candy bars.

**5 – 7 (1 point each) Consider the following method:**

public static int arrayMystery4(int[] list) {

int x = 0;

for (int i = 1; i < list.length; i++) {

int y = list[i] - list[0];

if (y > x) {

x = y;

}

}

return x;

}

For each array below, indicate what value would be returned if the method arrayMystery4 were called and passed that array as its parameter.

5. {5} \_\_\_\_\_\_\_

6. {4, 2, 10, 8} \_\_\_\_\_\_\_

7. {8, 2, 10, 4, 10, 9} \_\_\_\_\_\_\_

**8. (4 points) Write the entire Person class. You only need to provide the methods specified below**.

**Person class**

* Two fields: a full name and an age (integer).
* A 2-argument constructor that sets the name and age.
* A getter/accessor for the name
* A getter/accessor and setter/mutator for the age

**9. (6 points) Write the entire Student class. You only need to provide the methods specified below.**

**Student class**

The Student class inherits from the Person class (it is a subclass of Person). The Student class and has the following features:

* 2 additional fields: currentPoints and totalPoints (integers).
* A 2-argument constructor that sets name and age. The points fields should be initialized to 0.
* A method called getClassYear that will return the student’s class (a String) based on their age.

| **Age** | **Class** |
| --- | --- |
| 14 | Freshman |
| 15 | Sophomore |
| 16 | Junior |
| 17 or 18 | Senior |

* A setter/mutator method called addGrade that will increase the currentPoints and totalPoints fields. The method will be passed two integers: a grade and maxGrade.
  + For example: addGrade(85,100) will add 85 to currentPoints and add 100 to totalPoints.
* getGrade()which will return the Student’s current grade percentage
  + For example: if the student’s current points is 175 and the total points is 200, getGrade() should return 87.5.
* A toString() method which will return the student’s name, class, and current grade.

**Write the Student class on the next page.**

**Write the Student class – (cont. - more space if you need it)**

**Application of the Person and Student classes**

**10. (2 points) Complete the following method that will create and populate a lunch line of students.**

Your method will be passed an array of Strings with student names. It will return an array of Student objects (the lunch line).

Using the list of names, create Student objects and fill up an Array of Students called lunchLine. Generate a random age for each student in the range of 14 to 18.

**Example Input:**

studentList = {“Pat K”, “Bill B”, “Sam J”, “Ed A”, “Ned Z”}

**Example output list:**

| Student(“Pat K”, age=14) | Student(“Bill B”, age=16) | Student(“Sam J”, age=14) | Student(“Ed A”, age=15) | Student(“Ned Z”, age=18) |
| --- | --- | --- | --- | --- |

public static Student[] createLunchLine(String[] studentList){

}

**11. (3 points)** Every week, the students have extracurricular activities that prevent them from fully completing all their assignments, so they do as much as they can and hope for the best.

Write a method to add one random grade to each Student in the lunchLine. The random grade will be between the specified minimum grade and the maximum grade. For example: given a minScore of 60 and a maxScore of 100, the following scores \*could\* be generated: 90, 82, 63, 60, 100.

Each score would be added with the maxScore of 100.

public static void addRandomGrades(int minScore, int maxScore,

Student[] lunchLine) {

}

**12. (3 points)** The lunch line is always a hot mess… unless you’re an upperclassman with a decent grade average. Write a method that will swap the person at the front of the line with the student that has the highest grade of all the Juniors and Seniors. Display both the person who was displaced and the new leader of the line.

**BEFORE SWAP EXAMPLE**

| Student(“Pat K”, age=14, class=”Freshman”) | Student(“Bill B”, age=16, class=”Junior”) | Student(“Sam J”, age=14, class=”Freshman”) | Student(“Ed A”, age=15, class=”Sophomore”) | Student(“Ned Z”, age=18, class=”Senior”) |
| --- | --- | --- | --- | --- |

**AFTER SWAP EXAMPLE** (if Bill has a higher grade than Ned)

| Student(“Bill B”, age=16, class=”Junior”) | Student(“Pat K”, age=14, class=”Freshman”) | Student(“Sam J”, age=14, class=”Freshman”) | Student(“Ed A”, age=15, class=”Sophomore”) | Student(“Ned Z”, age=18, class=”Senior”) |
| --- | --- | --- | --- | --- |

public static void thereIsNoFreeLunch(Student[] lunchLine) {

}AP Computer Science \_\_\_\_\_\_\_ Name \_\_\_\_\_\_\_\_\_\_**SOLUTIONS**\_\_\_\_\_\_\_

Quiz: Inheritance, Polymorphism 60 points Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Period \_\_\_\_\_\_

Arrays, 2D Arrays

AP Quick Reference Sheet Allowed

**Multiple Choice (1 point each): Choose the best answer and circle it on your quiz.**

1. Assuming all variables are declared correctly, which of the following swaps two elements in an array of integers called numbers at the index of x and y?
2. numbers[x] = numbers[y];

numbers[y] = numbers[x];

1. int temp = numbers[x];

numbers[y] = numbers[x];

numbers[y] = temp;

1. int temp = numbers[x];

numbers[x] = numbers[y];

numbers[y] = temp;

1. I only
2. II only
3. III only
4. II and III
5. I, II, and III
6. What does the following code do? Assume list is an array of int values, temp is some previously initialized int value, and c is an int initialized to 0.

for (j = 0; j < list.length; j++)

if(list[j] < temp)

c++;

1. It finds the smallest value and stores it in temp
2. It finds the largest value and stores it in temp
3. It counts the number of elements equal to the smallest value in list
4. It counts the number of elements in list that are less than temp
5. It sorts the values in list to be in ascending order

**For questions 3 – 4, assume an int array, candy, stores the number of candy bars sold by a group of children where candy[j] is the number of candy bars sold by child j. Assume there are 12 children in all.**

3. Which of the following code segments could be used to compute the total number of bars sold by the children?

1. for (int j = 0; j < 12; j++)

sum += candy[j];

1. for (int j = 0; j < 12; j++)

candy[j] = sum;

1. for (int j = 0; j < 12; j++)

sum = candy[j];

1. for (int j = 0; j < 12; j++)

sum += [j];

1. for (int j = 0; j < 12; j++)

[j] += sum;

4. What does the following method do?

public int question5()

{

int value1 = 0;

int value2 = 0;

for (int j = 0; j < 12; j++){

if(candy[j] > value1)

{

value1 = candy[j];

value2 = j;

}

}

return value2;

}

1. It returns the total number of candy bars sold.
2. It returns the total number of children who sold 0 candy bars.
3. It returns the total number of children who sold more than 0 candy bars.
4. It returns the number of candy bars sold by the child who sold the most candy bars.
5. It returns the index of the child who sold the most candy bars.

**5 – 7 (1 point each) Consider the following method:**

public static int arrayMystery4(int[] list) {

int x = 0;

for (int i = 1; i < list.length; i++) {

int y = list[i] - list[0];

if (y > x) {

x = y;

}

}

return x;

}

For each array below, indicate what value would be returned if the method arrayMystery4 were called and passed that array as its parameter.

5. {5} \_\_\_0\_\_\_\_

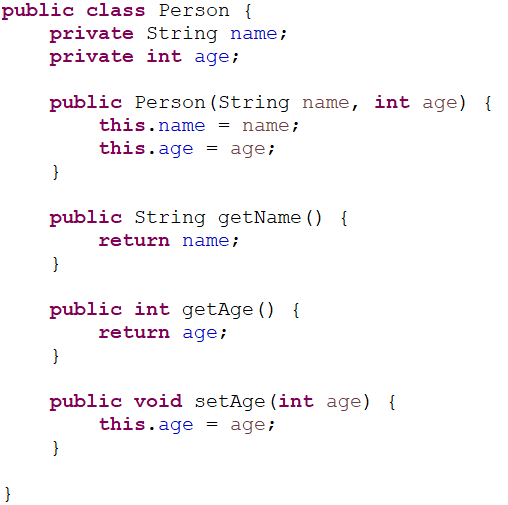
6. {4, 2, 10, 8} \_\_\_6\_\_\_\_

7. {8, 2, 10, 4, 10, 9} \_\_\_2\_\_\_\_

**8. (4 points) Write the entire Person class. You only need to provide the methods specified below**.

**Person class**

* Two fields: a full name and an age (integer).
* A 2-argument constructor that sets the name and age.
* A getter/accessor for the name
* A getter/accessor and setter/mutator for the age



4 points total:

* 1 point - fields declared private
* 1 point constructor
* 1 point getters
* 1 point setters

**9. (6 points) Write the entire Student class. You only need to provide the methods specified below.**

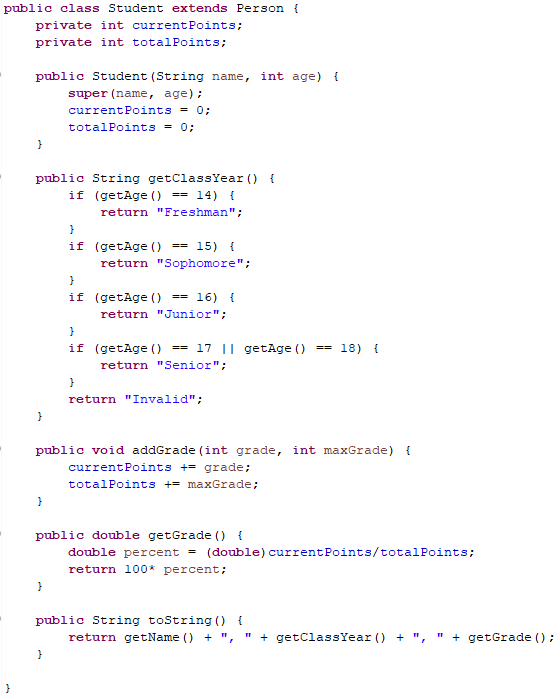
**Student class**

The Student class inherits from the Person class (it is a subclass of Person). The Student class and has the following features:

* 2 additional fields: currentPoints and totalPoints (integers).
* A 2-argument constructor that sets name and age. The points fields should be initialized to 0.
* A method called getClassYear that will return the student’s class (a String) based on their age.

| **Age** | **Class** |
| --- | --- |
| 14 | Freshman |
| 15 | Sophomore |
| 16 | Junior |
| 17 or 18 | Senior |

* A setter/mutator method called addGrade that will increase the currentPoints and totalPoints fields. The method will be passed two integers: a grade and maxGrade.
  + For example: addGrade(85,100) will add 85 to currentPoints and add 100 to totalPoints.
* getGrade()which will return the Student’s current grade percentage
  + For example: if the student’s current points is 175 and the total points is 200, getGrade() should return 87.5.
* A toString() method which will return the student’s name, class, and current grade.

****

6 points total:

* 1 point - additional fields created and declared private
* 1 point - constructor (includes call to the super)
* 1 point - getClass: 1/2 point for the header and return type; 1/2 point for the logic (must use the getter for age)
* 1 point - addGrade: 1/2 point for the header; 1/2 point for the logic
* 1 point - getGrade: 1/2 point for the header and return; 1/2 point for the math (casting to a double before division)
* 1 point - toString

**Application of the Person and Student classes**

**10. (2 points) Complete the following method that will create and populate a lunch line of students.**

Your method will be passed an array of Strings with student names. It will return an array of Student objects (the lunch line).

Using the list of names, create Student objects and fill up an Array of Students called lunchLine. Generate a random age for each student in the range of 14 to 18.

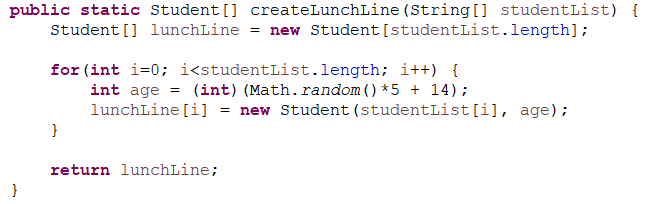
**Example Input:**

studentList = {“Pat K”, “Bill B”, “Sam J”, “Ed A”, “Ned Z”}

**Example output list:**

| Student(“Pat K”, age=14) | Student(“Bill B”, age=16) | Student(“Sam J”, age=14) | Student(“Ed A”, age=15) | Student(“Ned Z”, age=18) |
| --- | --- | --- | --- | --- |

public static Student[] createLunchLine(String[] studentList){



2 points total:

* 1/2 point for the return
* 1/2 point for creating the array
* 1/2 point for traversing the list and creating new Student objects
* 1/2 point for correctly using Math.random

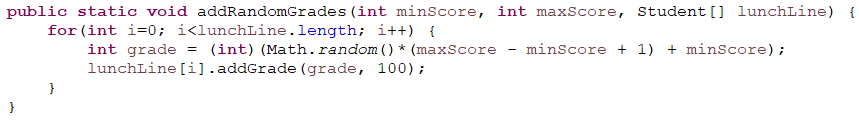
**11. (3 points)** Every week, the students have extracurricular activities that prevent them from fully completing all their assignments, so they do as much as they can and hope for the best.

Write a method to add one random grade to each Student in the lunchLine. The random grade will be between the specified minimum grade and the maximum grade. For example: given a minScore of 60 and a maxScore of 100, the following scores \*could\* be generated: 90, 82, 63, 60, 100.

Each score would be added with the maxScore of 100.

public static void addRandomGrades(int minScore, int maxScore,

Student[] lunchLine) {



3 points total:

* 1 point for correctly traversing the lunchLine
* 1 point for correctly using Math.random
* 1 point for calling addGrade to add a grade to each person in line

**12. (3 points)** The lunch line is always a hot mess… unless you’re an upperclassman with a decent grade average. Write a method that will swap the person at the front of the line with the student that has the highest grade of all the Juniors and Seniors. Display both the person who was displaced and the new leader of the line.

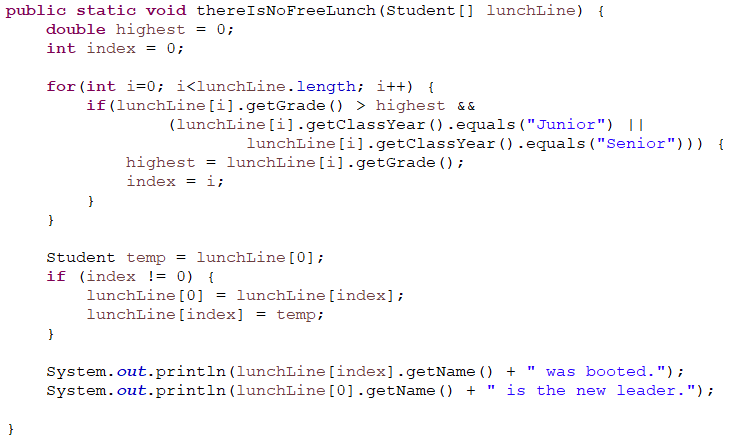
**BEFORE SWAP EXAMPLE**

| Student(“Pat K”, age=14, class=”Freshman”) | Student(“Bill B”, age=16, class=”Junior”) | Student(“Sam J”, age=14, class=”Freshman”) | Student(“Ed A”, age=15, class=”Sophomore”) | Student(“Ned Z”, age=18, class=”Senior”) |
| --- | --- | --- | --- | --- |

**AFTER SWAP EXAMPLE** (if Bill has a higher grade than Ned)

| Student(“Bill B”, age=16, class=”Junior”) | Student(“Pat K”, age=14, class=”Freshman”) | Student(“Sam J”, age=14, class=”Freshman”) | Student(“Ed A”, age=15, class=”Sophomore”) | Student(“Ned Z”, age=18, class=”Senior”) |
| --- | --- | --- | --- | --- |

public static void thereIsNoFreeLunch(Student[] lunchLine) {



3 points total:

* 1/2 point for traversing the lunchline
* 1 point for finding the highest grade (and making sure it’s a junior or senior)
* 1 point for the swap
* 1/2 point for displaying the students who were displaced