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

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

and 1D Arrays

AP Quick Reference Sheet Allowed

**Multiple Choice (1 points 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?

1. 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

1. 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 (int 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 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];

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

[j] += sum;

4. What does the following method do?

public int question4() {

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 points 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[0] - list[i];

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 with the full name and the age.

A getter/accessor for the full name

A setter/mutator and a getter/accessor for the age

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

**Employee class**

The Employee class inherits the Person class and has the following features:

* + Has one additional field; salary, which is a double.
  + A 3-argument constructor with the full name, the age, and the salary.
  + A setter/mutator method called addRaise that will increase the current salary by the given percent ( .10 = 10% raise).
    - For example, if the salary is $50,000 and the following call is made to addRaise(.10), the salary will be set to $55000.
  + getSalary()which will return the Employee’s current salary.
  + A toString() method which will return the employee’s name and current salary.

**Write the Employee class.**

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

**Application of the Person and Employee classes**

**10. (2 points) Complete the following method which will create and populate the cubicalChart of Employees.**

Your method will be passed an array of Strings with the employee names.

Generate a random age for each employee in the range of 18 to 75 and a random salary in the range of $25,000 and $145,000.

The cubical chart should be populated by rows**, but right to left, so the last column will be filled first**, followed by the next to last column, etc..

**Example**:

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

The CubicalChart will be populated in the following order

**Example:**

| Employee(“Sam J”, age=53, salary=52945) | Employee(“Ed A”, age=22, salary=135240) | Employee(“Ned Z”, age=72, salary=96000) | Employee(“Bill B”, age=28, salary=62950) | Employee(“Pat K”, age=35, salary=46234) |
| --- | --- | --- | --- | --- |

public static Employee[] createCubicalChart(String[] nameList) {

**11. (2 points)** Every year, employees get a raise based on performance and various other metrics. To test this feature, write a method which accepts the maximum percentage raise (.10 = 10%) and apply it to each employee in the cubical seating chart by a random percentage up to the maximum. For example, if the maxRaisePercent = .25, one employee might get a .08 raise, another might get a .23 raise, etc.

public static void addRaise(double maxRaisePercent, Employee[] seatingChart) {

}

**12. (3 points)** Most office spaces have prime seating spots, such as one that is next to a window with a nice view. Write a method that will swap the oldest employee with the employee in the first element.

**BEFORE SWAP EXAMPLE**

| Employee(“Sam J”, age=53, salary=52945) | Employee(“Ed A”, age=22, salary=135240) | Employee(“Ned Z”, age=72, salary=96000) | Employee(“Bill B”, age=28, salary=62950) | Employee(“Pat K”, age=35, salary=46234) |
| --- | --- | --- | --- | --- |

**AFTER SWAP EXAMPLE (this is what you should return)**

| Employee(“Ned Z”, age=72, salary=96000) | Employee(“Ed A”, age=22, salary=135240) | Employee(“Sam J”, age=53, salary=52945) | Employee(“Bill B”, age=28, salary=62950) | Employee(“Pat K”, age=35, salary=46234) |
| --- | --- | --- | --- | --- |

public static void ageHasItsAdvantantages(Employee[] seatingChart) {

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

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

Arrays, 2D Arrays

AP Quick Reference Sheet Allowed

**Multiple Choice (1 points 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 the at the index of x and y?

1. 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 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];

e. 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 points 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[0] - list[i];

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 – 15 (1 point each) Assuming that the following classes have been defined:**

public class A

{

public void method1()

{

System.out.print("A1 ");

}

public void method3()

{

System.out.print("A3 ");

}

}

public class B extends A

{

public void method3()

{

System.out.print("B3 ");

super.method3();

}

}

public class C extends B

{

public void method2()

{

System.out.print("C2 ");

method1();

}

}

public class D extends B

{

public void method1()

{

System.out.print("D1 ");

super.method1();

}

public void method3()

{

System.out.print("D3 ");

super.method3();

}

}

And assuming the following variables have been defined:

A var1 = new C();

B var2 = new B();

A var3 = new D();

D var4 = new D();

Object var5 = new A();

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement causes an error, fill in the right-hand column with either the phrase “compiler error” or “runtime error” to indicate when the error would be detected.

Statement Output

8. var1.method1(); \_\_\_\_\_\_\_\_\_\_ A1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

9. var2.method3(); \_\_\_\_\_\_\_\_\_\_\_\_\_B3 A3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

10. var1.method2(); \_\_\_\_\_\_\_\_\_\_\_\_Compile Error\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

11. ((B)var1).method2(); \_\_\_\_\_\_\_\_\_ Compile Error \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

12. ((D)var5).method1(); \_\_\_\_\_\_\_\_\_\_\_run time error\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

13. ((D)var3).method3(); \_\_\_\_\_\_\_\_\_\_ D3 B3 A3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**14. (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 with the full name and the age.

A getter/accessor for the full name

A setter/mutator for the age

**public** **class** Person {

**private** String fullName;

**private** **int** age;

**public** Person(String fullName, **int** age) {

**this**.fullName = fullName;

**this**.age = age;

}

**public** String getFullName() {

**return** fullName;

}

**public** **void** setFullName(String fullName) {

**this**.fullName = fullName;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**19. (6 points) Write the entire Employee class. You only need to provide the methods specified below.**

**Employee class**

The Employee class inherits the Person class and has the following features:

* + Has one additional field; salary, which is a double.
  + A 3-argument constructor with the full name, the age, and the salary.
  + A setter/mutator method called addRaise that will increase the current salary by the given percent (.10 = 10% raise).
    - For example, if the salary is $50,000 and the following call is made to addRaise(.10), the salary will be set to $55000.
  + getSalary()which will return the Employee’s current salary.
  + A toString() method which will return the employee’s name and current salary.

**Write the Employee class.**

**Application of the Person and Employee classes**

20. (2 points) Complete the following method which will create and populate the cubicalChart of Employee.

Your method will be passed an array of Strings with the employee names.

Generate a random age for each employee in the range of 18 to 75 and a random salary in the range of $25,000 and $145,000.

The cubical chart should be populated by rows, but right to left, so the last column will be filled first, followed by the next to last column, etc..

**Example**: rows = 2

columns = 3

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

The CubicalChart will be populated in the following order

**Example:**

| Employee(“Sam J”, age=53, salary=52945) | Employee(“Ed A”, age=22, salary=135240) | Employee(“Ned Z”, age=72, salary=96000) | Employee(“Bill B”, age=28, salary=62950) | Employee(“Pat K”, age=35, salary=46234) |
| --- | --- | --- | --- | --- |

public static Employee[] createCubicalChart(String[] nameList) {

21. (2 points) Every year, employees get a raise based on performance and various other metrics. To test this feature, write a method which accepts the maximum percentage raise (.10 = 10%) and apply it to each employee in the cubical seating chart by a random percentage up to the maximum. For example, if the maxRaisePercent = .25, one employee might get a .08 raise, another might get a .23 raise, etc.

**public** **static** **void** addRaise(**double** maxRaisePercent, Employee[] seatingChart) {

**for**(**int** i=0; i<seatingChart.length; i++) {

**double** raise = Math.*random*()\*maxRaisePercent;

seatingChart[i].addRaise(raise);

}

}

}

22. (3 points) Most office spaces have prime seating spots, such as one that is next to a window with a nice view. Write a method that will swap the employee with the highest salary with the employee in the first element.

**BEFORE SWAP EXAMPLE**

| Employee(“Sam J”, age=53, salary=52945) | Employee(“Ed A”, age=22, salary=135240) | Employee(“Ned Z”, age=72, salary=96000) | Employee(“Bill B”, age=28, salary=62950) | Employee(“Pat K”, age=35, salary=46234) |
| --- | --- | --- | --- | --- |

**AFTER SWAP EXAMPLE (this is what you should return)**

| Employee(“Ned Z”, age=72, salary=96000) | Employee(“Ed A”, age=22, salary=135240) | Employee(“Sam J”, age=53, salary=52945) | Employee(“Bill B”, age=28, salary=62950) | Employee(“Pat K”, age=35, salary=46234) |
| --- | --- | --- | --- | --- |

**public** **static** **void** ageHasItsAdvantantages(Employee[] seatingChart) {

**int** maxi=0;

**for** (**int** i=0;i<seatingChart.length;i++) {

**if**(seatingChart[i].getAge()>seatingChart[maxi].getAge()) {

maxi = i;

}

}

Employee tempEmployee = seatingChart[0];

seatingChart[0] = seatingChart[maxi];

seatingChart[maxi]=tempEmployee;

}