AP Java Final Review (not collected) (solutions available at apcslowell.github.io)

12102021

1. What is the output of the following program?

```
int[] nums1 = {1,2,3};
int[] nums2 = {}; //empty!
public void setup()
{
  System.out.println(mystery(nums1)+mystery(nums2));
}
public int mystery(int[] arr)
{
  int x = 0;
  for(int k = 0; k < arr.length; k = k + 1)
    x = x + arr[k];
  return x;
}
```

2. For the following lines of code, label each line as a declaration, initialization or both.

```
int num;
num = 4;
SpaceShip bob = new SpaceShip();
String word = new String("Hello");
double dNum;
dNum = 3.14;
```

3. Syntax errors can be thought of as spelling mistakes. Syntax errors are detected before a Java program runs. Exceptions are problems of logic that are detected when a program is running. For each of the following problems label them as an Exception or Syntax Error:

```
A. int num = 5/0;
B. int[] arr = {1,2,3}; System.out.println(arr[3]);
C. system.out.println("Hello World!");
D. String bob; System.out.println(bob.length());
E. String bob2 = "3"; System.out.println((int)bob2);
F. System.out.println("What problem?")
```

4. The following code compiles without error, but when it runs a **`ClassCastException`** is generated. Circle the one line of code that causes the **`Exception.`**

```
public void setup(){
        One first = new One();
        Two second = new Two();
        System.out.println(first.getNum());
        System.out.println(second.getNum());
        System.out.println(((One)second).getNum());
        System.out.println(((Two)first).getNum());
}
class One{
        protected int myNum;
        public One(){myNum = 1;}
        public int getNum(){return myNum;}
}
class Two extends One{
        public Two(){myNum = 2;}
}
```

5. What is the output of the following program?

```
System.out.println(1/2*3);
System.out.println((double)(1/2)*3);
System.out.println((double)1/2*3);
System.out.println(1/(double)2*3);
```

6. Write ***two*** different pieces of code that both correctly declare and initialize an array with the elements {0,1,2,3,4,5}

7. The following program segment is intended to find the index of the first negative integer an array.

```
int nI = 0;
while(naArray[nI] >= 0){
   nI++;
}
int nLocation = nI;
```

This segment will work as intended
A. Always
B. Never
C. Whenever naArray contains at least one negative integer
D. Whenever naArray contains at least one nonnegative integer
E. Whenever naArray contains no negative integers

8. What is the output of the following code segement?

```
ArrayList <Integer> theList = new ArrayList <Integer>();
for(int nI = 1; nI < 20; nI=nI*2){
     theList.add(nI);
}
Integer A = theList.get(3);
System.out.println(A);
Integer B = theList.get(1);
System.out.println(B);
int nC = theList.get(2);
System.out.println(nC);
```

9. Refer to the following declarations:

```
String[] colors = {"red", "green", "black"};
ArrayList <String> colorList = new ArrayList<String>();
```

Which of the following correctly assigns the elements of the **`colors`** array to **`colorList`**? The final ordering of the colors in **`colorList`** should be the same as the **`colors`** array.

```
I for(int i = 0; i<colors.length; i++){
   colorList.add(i,colors.get(i));
  }
II for(int i =0; i< colors.length; i++){
    colorList.add(colors[i]);
   }
III for(int i = colors.length-1; i>= 0; i--){
     colorList.add(i, colors[i]);
    }
```

A. I only B. II only C. III only D. II and III only E. I, II, and III

10. What is the primary advantage of an **`ArrayList`** as opposed to an array?

11. Name a disadvantage of using an **`ArrayList`** rather than an array.

12. Consider the following program.

```
for(int k=0; k<7;k++)
  for(int j = 0; j<3; j++)
    System.out.print("*");
```

How many stars are output when this code segment is executed?

13. Fill in the blanks in the following block of code so that it displays the sum of all integers from **1** to **nNum.** Assume that **nNum** has been initialized to a positive integer.

```
int nSum = 0;
int nInt = nNum;
while (nInt != 0)
{
        nSum = nSum + nInt;

        ____________________;
}

System.out.println(____________________);
```

14. What is the output of the following program? What would the output be if the **||** was changed to an **&&**?

```
WhatsIt one = new WhatsIt(1);
WhatsIt two = new WhatsIt(0);
if(one.getNum() < 3 ||  3/two.getNum() < 1){
        System.out.println("Success");
}

class WhatsIt{
   private int myNum;
   public WhatsIt(int nNum){myNum = nNum;}
   public int getNum(){return myNum;}
}
```

15. What is the output of the following program?

```
System.out.println((double)(7 * 9 / 2));
System.out.println((double)7 * ((double)9) / 2);
System.out.println((double)7 * (double)(9 / 2));
System.out.println((double)7 * (double)9 / 2);
```

16. What is the output of the following 3 loops?

```
int i = 0;
while(i < 2){
    System.out.println(i);
    i++;
}
```

```
int j = 0;
while(j < 2){
    j++;
    System.out.println(j);
}
```

```
int k = 0;
while(k < 2){
    k++;
}
System.out.println(k);
```

Questions 17 – 19 use the following program.

```
public void setup() {
    BankAccount s = new SavingsAccount(100,.10);
    ((SavingsAccount)s).addInterest(); //question #18
    System.out.println(s.writeBalance());
    BankAccount c = new CheckingAccount(100);
    c.withdraw(60);
    System.out.println(c.writeBalance());
}
class BankAccount{
   protected double myBalance;
   public BankAccount(){myBalance = 0;}
   public BankAccount(double balance){myBalance = balance;}
   public void deposit(double amount){myBalance += amount;}
   public void withdraw(double amount){myBalance -= amount;}
   public double getBalance(){return myBalance;}
   public String writeBalance(){/* implementation code */} //question #17
}
class SavingsAccount extends BankAccount{
   private double myInterestRate;
   public SavingsAccount(){ myBalance = 0; myInterestRate = 0; }
   public SavingsAccount(double balance, double rate){
        myBalance = balance;
        myInterestRate = rate;
   }
   public void addInterest(){ myBalance*=(1+myInterestRate); } //Add interest to balance
}
class CheckingAccount extends BankAccount{
   private static final double FEE = 2.0;
   private static final double MIN_BALANCE = 50.0;
   public CheckingAccount(double balance){ myBalance = balance; }
   /* FEE of $2 deducted if withdrawal leaves balance less than MIN_BALANCE.
    * Allows for negative balance. */
   public void withdraw(double amount){
       myBalance -= amount;
       if(myBalance < MIN_BALANCE)
         myBalance-=2;
    }
}
```

17. Which of the following could be used as `/* implementation code */` for the `writeBalance()` function?

```
I. return "Balance: $" + getBalance();
II. return "Balance: $" + balance;
III. return "Balance: $" + myBalance;
```

a. I only b. II only c. III only d. I and III only e. I, II and III

18. Why is it necessary to cast `s` as a `SavingsAccount` in problem 17?

19. Assuming `writeBalance` has a correct implementation, what is the output of the program?

20. What will happen when the following program is run?

```
int [] nums = {1,2,3};
System.out.println(nums[3]);
```

a. The compiler issues a warning
b. The compiler reports a syntax error
c. The program will display a seemingly random number.
d. The program freezes up and stops responding to the user).
e. The program throws an `ArrayIndexOutOfBoundsException`.

21. What is the output of the following program?

```
System.out.println( 1 != 0 || 1 != 1 || 1 != 2);
System.out.println( 1 != 0 && 1 != 1 && 1 != 2);
```

Questions 22 and 23 use the following incomplete class definitions

```
class Mammal{
   private int mySize;
   public Mammal(){ ...}
   public Mammal(int size){mySize = size;}
   public int getSize(){ . . . }
}

class Wolf extends Mammal{
   public Wolf() { ... }
   public void speak(){ .... }
}

class Dog extends Wolf{
   private int myDogLicense;
   public Dog(int size, int license){myDogLicense = license;}
   public int getLicense(){ ... }
}

class Cat extends Mammal{
   public Cat(){ ... }
}
```

22. Which classes have a default (no argument) constructor? How many non-constructor methods can be invoked (used) by a **Dog** object?

23. A program to test the **Mammal**, **Wolf**, **Dog** and **Cat** classes has these declarations:

```
Mammal a = new Mammal();
Mammal b = new Wolf();
Mammal c = new Dog(37, 12345678);
Mammal d = new Cat();
```

For the following lines of code, circle those that will generate an error. Mark an X next to those that will generate a **ClassCastException**.

```
a.getSize();
a.speak();
b.getSize();
b.speak();
((Wolf)b).speak();
c.getSize();
c.speak();
((Wolf)c).speak();
((Dog)c).speak();
d.getSize();
d.speak();
((Wolf)d).speak();
((Cat)d).speak();
```

24. Consider the following class definitions.

```
class Apple{
  public void printColor(){System.out.print("Red");}
}
class GrannySmith extends Apple{
  public void printColor(){System.out.print("Green");}
}
class Jonagold extends Apple{
  // no methods defined
}
```

The following statement appears in a method in another class.

```
someApple.printColor();
```

Under which of the following conditions will the statement print "Red" ?

I. When **someApple** is an object of type **Apple**

II. When **`someApple`** is an object of type **`GrannySmith`**
III. When **`someApple`** is an object of type **`Jonagold`**

A. I only B. I and II only C. I and III only D. II and III only E. I, II, and III

Questions 25 – 28 use the following program. Line numbers are included for reference:

```
1.
2.  public void setup() {
3.    BaseClass[] array = {new BaseClass(),new DerivedClass1(),new DerivedClass2()};
4.    for (int nI = 0; nI < array.length; nI++)
5.      array[nI].mystery(nI);
6.    ((DerivedClass2)array[2]).mystery();
7.    for (BaseClass thing : array)
8.      System.out.print(thing.getLetter() + " ");
9.  }
10. class BaseClass {
11.   protected char myLetter;
12.   public BaseClass() {
13.     myLetter = 'a';
14.   }
15.   public void mystery(int nNum) {
16.     myLetter=char(myLetter + nNum); //hint 'a' + 1 is 'b'
17.   }
18.   public char getLetter() {
19.     return myLetter;
20.   }
21. }
22. class DerivedClass1 extends BaseClass {
23.   public DerivedClass1() {
24.     myLetter = 'b';
25.   }
26.   public void mystery(int nNum) {
27.     myLetter+=(2*nNum); //hint 'a'+=(2*1) is 'c'
28.   }
29. }
30. class DerivedClass2 extends DerivedClass1 {
31.   public DerivedClass2() {
32.     myLetter = 'c';
33.   }
34.   public void mystery() {
35.     myLetter++; //hint 'a'++ is 'b'
36.   }
37. }
```

25. Which line of code is an example of *polymorphism*?

26. Which line of code is an example of *overloading*?

27. Which line of code is an example of *overriding*?

28. What is the output of the program?

29. Which of the following would be correct solutions for a has77 function that given an array of ints, return true if the array contains two 7's next to each other, or there are two 7's separated by one element, such as with {7, 1, 7}.

```
public boolean has77(int[] nums) {
  for(int i = 0; i < nums.length-1; i++)
    if(nums[i] == 7 && nums[i+1] == 7)
      return true;
    else ifnums[i] == 7 && nums[i+2]==7)
      return true;
  return false;
}
```

```
public boolean has77(int[] nums) {
  for(int i = 0; i < nums.length-1; i++)
    if(nums[i] == 7 && nums[i+1] == 7)
      return true;
  for(int i = 0; i < nums.length-1; i++)
    if( nums[i] == 7 && nums[i+2]==7)
      return true;
  return false;
}
```

```
public boolean has77(int[] nums) {
  for(int i = 0; i < nums.length-1; i++)
    if(nums[i] == 7 && nums[i+1] == 7)
      return true;
    else if(i+2 < nums.length &&
        nums[i] == 7 && nums[i+2]==7)
      return true;
  return false;
}
```

```
public boolean has77(int[] nums) {
  for(int i = 0; i < nums.length-1; i++)
    if(nums[i] == 7 && nums[i+1] == 7)
      return true;
  for(int i = 0; i < nums.length-2; i++)
    if( nums[i] == 7 && nums[i+2]==7)
      return true;
  return false;
}
```

30. What is the output of the following program?

```
boolean bX= false;
for (int i = 0; i < 10; i++){
  if (bX) //same as if(bX == true)
    System.out.print(i + ", ");
  bX = !bX; //bX is assigned the opposite value
}
```

31. What is the output of the following program?

```
int a=10;
int b=10;
int c=10;
if(a==10){
  if(b==0){
    c = c / 2;
  }
}
else{
  c++;
}
System.out.println(c);
```

32. Consider the following class declarations.

```
class Dog{
  private String name;
  public Dog(){name = "NoName";}
}
```

```
class Poodle extends Dog{
  private String size;
  public Poodle(String s) {size = s;}
}
```

The following statement appears in a method in another class.

```
Poodle myDog = new Poodle("toy");
```

Which of the following best describes the result of executing the statement?

A. The **Poodle** variable **myDog** is instantiated as a **Poodle**. The instance variable **size** is initialized to **"toy"**. The instance variable **name** is not assigned a value.
B. The **Poodle** variable **myDog** is instantiated as a **Poodle**. The instance variable **size** is initialized to **"toy"**. An implicit call to the no-argument **Dog** constructor is made, initializing the instance variable name to **"NoName"**.
C. The **Poodle** variable **myDog** is instantiated as a **Poodle**. The instance variable **size** is initialized to **"toy"**. An implicit call to the no-argument **Dog** constructor is made, initializing the instance variable name to **"toy"**.
D. A runtime error occurs because **super** is not used to call the no-argument **Dog** constructor.
E. A runtime error occurs because there is no one-argument **Dog** constructor.

33. Which of the following would be correct solutions for a **withoutAny3sList** function that given an **ArrayList** of **Integer**s, returns an **ArrayList** after removing all elements with a value equal to 3. (Hint: There are multiple correct solutions)

```
public ArrayList<Integer> withoutAny3sList(ArrayList<Integer> nums){
  for(int i = 0; i < nums.size(); i++)
    if(nums.get(i) == 3)
      nums.remove(i);
  return nums;
}
```

```
public ArrayList<Integer> withoutAny3sList(ArrayList<Integer> nums){
  for(int i = nums.size()-1; i >= 0; i--)
    if(nums.get(i) == 3)
      nums.remove(i);
  return nums;
}
```

```
public ArrayList<Integer> withoutAny3sList(ArrayList<Integer> nums){
  for(int i = 0; i < nums.size(); i++)
    if(nums.get(i) == 3){
      nums.remove(i);
      i--;
    }
  return nums;
}
```

```
public ArrayList<Integer> withoutAny3sList(ArrayList<Integer> nums){
  ArrayList <Integer> answer = new ArrayList <Integer> ();
  for(int i = 0; i < nums.size(); i++)
    if(nums.get(i) == 3)
      answer.add(nums.get(i));
  return answer;
}
```

```
public ArrayList<Integer> withoutAny3sList(ArrayList<Integer> nums){
  int i = 0;
  while(i < nums.size())
    if(nums.get(i) == 3)
      nums.remove(i);
    else
      i++;
  return nums;
}
```

```
public ArrayList<Integer> withoutAny3sList(ArrayList<Integer> nums){
  ArrayList <Integer> answer = new ArrayList <Integer> ();
  for(int i = 0; i < nums.size(); i++)
    if(nums.get(i) != 3)
      answer.add(nums.get(i));
  return answer;
}
```