SPOILER Link to solutions and explanations:
https://docs.google.com/document/d/12McLdcV26jE9A9yZEAMjE9w_UE2hRRWfHg2PJXH9s30/edit?usp=sharing

1. Which of the following statements will create a string "Review Session"?
   a) String s = "Review Session";
   b) String s = new String("Review Session");
   c) String s; s = "Review Session";
   d) String s = "Review "; s += "Session";
   e) all of the above
   f) none of the above

2. What does the following statements print to console?

```
String s1 = "Review Session";
String s2 = s1;
String s3 = s2;
s2 += " for Final";
System.out.println(s1 == s2);
System.out.println(s1 == s3);
System.out.println(s2 == s3);
```

   a) false\nfalse\nfalse\n
   b) false\ntrue\nfalse\n
   c) true\nfalse\ntrue\n
   d) true\ntrue\ntrue\n
   e) none of the above

3. Which of the following gives the number of characters in String variableName?
   a) variableName.length
   b) variableName.length()
   c) variableName.size
   d) variableName.size()
   e) none of the above

4. Which of the following accesses the (i + 1)th character i in String variableName?
   a) variableName[i]
   b) variableName(i)
   c) variableName.get(i)
   d) variableName.charAt(i)
   e) variableName.indexAt(i)
   f) none of the above

5. Which of the following gives the number of ints in int[] variableName?
   a) variableName.length
   b) variableName.length()
   c) variableName.size
   d) variableName.size()
   e) none of the above

6. Which of the following accesses the (i + 1)th int in int[] variableName?
   a) variableName[i]
   b) variableName(i)
   c) variableName.get(i)
   d) variableName.intAt(i)
   e) variableName.indexAt(i)
   f) none of the above

7. Which of the following gives the number of Objects in Object[] variableName?
   a) variableName.length
   b) variableName.length()
   c) variableName.size
   d) variableName.size()
   e) none of the above

8. Which of the following gives the number of Objects in ArrayList<Object> variableName?
   a) variableName.length
   b) variableName.length()
   c) variableName.size
   d) variableName.size()
   e) none of the above

9. Which of the following accesses the (i + 1)th Object in ArrayList<Object> variableName?
   a) variableName[i]
   b) variableName(i)
   c) variableName.get(i)
   d) variableName.objectAt(i)
   e) variableName.indexAt(i)
   f) none of the above

10. What does the following statements print to console?

```
public class Main {
   public static void giveMeFive(int input) {
      input = 5;
   }
   public static void main(String[] args) {
      int input = 0;
      giveMeFive(input);
      System.out.println(input);
   }
}
```

   a) 0
   b) 5
   c) void
   d) null
   e) none of the above

11. What does the following statements print to console?

```
public class Main {
   public static int giveMeFive(int input) {
      return input = 5;
   }
   public static void main(String[] args) {
      int input = 0;
      System.out.println(giveMeFive(input) + " " + input);
   }
}
```

a) 0 0
b) 0 5
c) 5 0
d) 5 5
e) none of the above

12. What does the following statements print to console?

```
public class Main {
   public static void giveMeFive(Integer input) {
      input = 5;
   }
   public static void main(String[] args) {
      Integer input = 0;
      giveMeFive(input);
      System.out.println(input);
   }
}
```

a) 0
b) 5
c) void
d) null
e) none of the above

13. What does the following statements print to console?

```
public class Main {
   public static void giveMeFive(int[] input) {
      input[0] = 5;
   }
   public static void main(String[] args) {
      int[] input = {0}; // same as "int[] input = new int[] {0};"
      giveMeFive(input);
      System.out.println(input[0]);
   }
}
```

a) 0
b) 5
c) void
d) null
e) none of the above

14. What does the following statements print to console?

```
import java.util.ArrayList;
public class Main {
   public static void start(ArrayList<Integer> input) {
      input.add(5);
   }

   public static void main(String[] args) {
      ArrayList<Integer> input = new ArrayList<Integer>();
      start(input);
      System.out.println(input);
   }
}
```

a) name of the class (java.util.ArrayList) followed by the object's hexadecimal address in memory
b) 5
c) []
d) [5]
e) none of the above

15. What does the following statements print to console?

```
public class Main {
   int x = 1;

   public void start(int x) {
      x = 2;
      System.out.print(x + " " + this.x + " ");
   }

   public static void main(String[] args) {
      int x = 3;
      new Main().start(x);
      System.out.println(x);
   }
}
```

a) 1 2 3
b) 1 2 2
c) 2 1 1
d) 2 1 3
e) none of the above

16. What does the following statements print to console?

```
public class Main {
   int x = 1;
   int y = 3;

   public void first(int x) {
      int temp = x;
      y = x / 2;
      x = temp;
   }

   void second(int x, int z) {
      x = y + 2;
      first(x);
   }

   public void start() {
      int x = 2;
      y++;
      first(y);
      second(y, x);
      System.out.println(this.y);
   }

   public static void main(String[] args) {
      new Main().start();
   }
}
```

a) 1
b) 2
c) 4
d) 6
e) none of the above

17. What does the following statements print to console?

```
public class Main {
   int x = 1;
   int thisX = 2;
   Integer thatX = 3;
   public Main() {
      x = 4;
      this.thisX = 5;
      Integer thatX = 6;
   }
   public void method(Integer x) {
      int temp = this.thisX;
      this.thisX = x;
      x = temp;
      System.out.print("x = " + x + "; this.thisX = " + this.thisX + "; thatX = " + thatX);
   }
   public static void main(String[] args) {
      Main main = new Main();
      Integer thatOtherX = 7;
      main.method(thatOtherX);
      System.out.println("; thatOtherX = " + thatOtherX);
   }
}
```

a) x = 5; this.thisX = 7; thatX = 3; thatOtherX = 6
b) x = 5; this.thisX = 7; thatX = 3; thatOtherX = 7
c) x = 5; this.thisX = 7; thatX = 6; thatOtherX = 6
d) x = 5; this.thisX = 7; thatX = 6; thatOtherX = 7
e) none of the above

18. What does the following statements print to console?

```
java.util.ArrayList<Character> arrayList = new
   java.util.ArrayList<Character>();
arrayList.add('a');
arrayList.add('c');
arrayList.add('c');
arrayList.add('d');
arrayList.add('c');
for (int i = 0; i < arrayList.size(); ++i) {
   if (arrayList.get(i) == 'c') {
      arrayList.remove(i);
   }
}
System.out.println(arrayList);
```

a) [a, d]
b) [a, c, d]
c) [a, c, d, c]
d) name of the class (java.util.ArrayList) followed by the object's hexadecimal address in memory
e) none of the above

19. What does the following statements print to console?

```
class Animal {
   public Animal () {
      System.out.print("animalConstructor ");
   }
}
class Dog extends Animal {
   public Dog () {
      System.out.print("dogConstructor ");
   }
}
public class Main {
   public static void main(String[] args) {
      Animal dogAnimal = new Dog();
   }
}
```

a) dogConstructor
b) animalConstructor
c) animalConstructor dogConstructor
d) dogConstructor animalConstructor
e) none of the above

20. What does the following statements print to console?

```
class Animal {
}
class Dog extends Animal {
}
public class Main {
   public static void main(String[] args) {
      Animal dogAnimal = new Dog();
      if (dogAnimal instanceof Dog) {
            System.out.print("dog ");
      }
      if (dogAnimal instanceof Animal) {
            System.out.print("animal ");
      }
   }
}
```

a)
b) dog
c) animal
d) dog animal
e) none of the above

21. What does the following statements print to console?

```
class Animal {
   public boolean equals(Object other) { return false; }
}
class Dog extends Animal {
   public boolean equals(Animal other) { return true; }
}
public class Main {
   public static void main(String[] args) {
      Animal dogAnimal = new Dog();
      System.out.println(dogAnimal.equals(new Animal()));
   }
}
```

a) true
b) false
c) null
d) void
e) none of the above

22. What does the following statements print to console?

```
class Animal {
   public boolean equals(Object other) { return false; }
}
class Dog extends Animal {
   public boolean equals(Object other) { return true; }
}
public class Main {
   public static void main(String[] args) {
      Animal dogAnimal = new Dog();
      System.out.println(dogAnimal.equals(new Animal()));
   }
}
```

a) true
b) false
c) null
d) void
e) none of the above

23. What does the following statements print to console?

```
class Animal {
   public int height = 1;
}
class Dog extends Animal {
   public int height = 2;
}
public class Main {
   public static void main(String[] args) {
      Animal dogAnimal = new Dog();
      Animal secondDogAnimal = dogAnimal;
      secondDogAnimal.height = 3;
      System.out.println(dogAnimal.height + " " + secondDogAnimal.height);
   }
}
```

a) 1 3
b) 2 2
c) 2 3
d) 3 3
e) none of the above

24. What does the following statements print to console?

```
class Animal {
   public int height = 1;
   public void grow() {
      height++;
   }
}
public class Main {
   public static void main(String[] args) {
      Animal animal = new Animal();
      Animal secondAnimal = animal;
      secondAnimal.grow();
      System.out.println(animal.height + " " + secondAnimal.height);
   }
}
```

a) 1 1
b) 1 2
c) 2 1
d) 2 2
e) none of the above

25. What does the following statements print to console?

```
class Animal {
   public int height = 1;
   public void grow() {
      height++;
   }
}
public class Main {
   public static void main(String[] args) {
      Animal animal = new Animal();
      Animal secondAnimal = new Animal();
      secondAnimal.grow();
      System.out.println(animal.height + " " + secondAnimal.height);
   }
}
```

a) 1 1
b) 1 2
c) 2 1
d) 2 2
e) none of the above

26. What does the following statements print to console?

```
class Animal {
   public static int height = 1;
   public void grow() {
      height++;
   }
}
public class Main {
   public static void main(String[] args) {
      Animal animal = new Animal();
      Animal secondAnimal = new Animal();
      secondAnimal.grow();
      System.out.println(animal.height + " " + secondAnimal.height);
   }
}
```

a) 1 1
b) 1 2
c) 2 1
d) 2 2
e) none of the above

27. An object is an instance of a

a) class
b) program
c) method
d) data
e) none of the above

28. Multiple true/false question: Which of the followings are true about `System.out.println(...)` statement?
    a) `System` is a class that we need to define
    b) `out` is a private reference field in `System`
    c) `out` is a static reference field in `System`
    d) `println` is a method belongs to `System`
    e) `println` is an overloaded method
    f) When passing a reference variable to `println`, it prints the variable class followed by @ and the object's address in memory
    g) Whatever passed to the `println` goes to a buffer first

29. Multiple true/false question: Which of the followings are true about System.in.read() statement?
    a) It can be used to read strings
    b) It can cause the program to throw `FileNotFoundException`
    c) It reads 8-bits ASCII value when available
    d) `System.in` can be used as an argument to a `Scanner` object to read strings

30. What does the following statements print to console?

```
try {
   System.out.print("try ");
   throw new ArithmeticException();
} catch (ArithmeticException excpt) {
   System.out.print("catchArithmetic ");
   throw new ArrayIndexOutOfBoundsException();
} catch (ArrayIndexOutOfBoundsException excpt) {
   System.out.print("catchBounds ");
} finally {
   System.out.print("finally ");
}
```

    a) try catchArithmetric catchBounds
    b) try catchArithmetric catchBounds finally
    c) try catchArithmetric
    d) try catchArithmetric finally
    e) none of the above

31. What does the following statements print to console?

```
try {
   try {
      System.out.print("try ");
      throw new ArithmeticException();
   } catch (ArithmeticException excpt) {
      System.out.print("catchArithmetic ");
      throw new ArrayIndexOutOfBoundsException();
   } catch (ArrayIndexOutOfBoundsException excpt) {
      System.out.print("catchBounds ");
   } finally {
      System.out.print("finally ");
   }
} catch (Exception excpt) {
   System.out.print("catchExcpt ");
}
```

a) try catchArithmetric catchBounds catchExcpt
b) try catchArithmetric catchBounds catchExcpt finally
c) try catchArithmetric catchBounds finally
d) try catchArithmetric catchBounds finally catchExcpt
e) try catchArithmetric catchExcpt
f) try catchArithmetric catchExcpt finally
g) try catchArithmetric finally catchExcpt
h) none of the above

32. Which of the following lines of code to successfully write "something" to a file given PrintWriter printWriter that has been declared and initialized.
a) printWriter("something");
b) printWriter.println("something");
c) printWriter.out.println("something");
d) printerWriter.writeln("something");
e) none of the above

33. True or False: If private instance variables were made public, we would not need the *get* (accessor) and *set* (mutator) methods.
a) True
b) False

34. Which accessSpecifier for height will compile?

```
class Animal {
   [accessSpecifier] int height = 1;
}
public class Main {
   public static void main(String[] args) {
      System.out.println(new Animal().height);
   }
}
```

a) public
b) public and protected
c) public, protected, and no modifier
d) public, protected, no modifier, and private

35. Which accessSpecifier for height will compile?

Animal.java:

```
package firstPackage;

public class Animal {
   [accessSpecifier] int height = 1;
}
```

Dog.java:

```
package secondPackage;
import firstPackage.Animal;

public class Dog extends Animal {
   public Dog() {
         height = 2;
   }
}
```

Main.java:

```
package firstPackage;
import secondPackage.Dog;

public class Main {
   public static void main(String[] args) {
         System.out.println(new Dog().height);
   }
}
```

a) public
b) public and protected
c) public, protected, and no specifier
d) public, protected, no specifier, and private
e) none of the above

36. What does the following statements print to console?

```
abstract class Animal {
   abstract void makeSound();
}
class Dog extends Animal {
   @Override
   void makeSound() {
      System.out.print("woof ");
   }
}
class Cat extends Animal {
   @Override
   void makeSound() {
      System.out.print("meow ");
   }
}
public class Main {
   public static void main(String[] args) {
      Dog dog = new Dog();
      Cat cat = new Cat();
      Animal dogAnimal = dog;
      dog.makeSound();
      cat.makeSound();
      dogAnimal.makeSound();
   }
}
```

a) will not compile
b) meow meow meow
c) woof meow meow
d) woof meow woof
e) none of the above

37. True or False: The following will compile and run despite instantiating an abstract class.

```
abstract class Animal {
   abstract void makeSound();
}
public class Main {
   public static void main(String[] args) {
      Animal animal = new Animal();
   }
}
```

a) True
b) False

38. True or False: The following will compile despite extending an abstract class without implementing the abstract method.

```
abstract class Animal {
   abstract void makeSound();
}
class CookieMonster extends Animal {
   void eat() {
      System.out.print("nom nom nom ");
   }
}
```

a) True
b) False

39. True or False: The following will compile despite creating an abstract class that extends another abstract class without implementing the abstract method.

```
abstract class Animal {
   abstract void makeSound();
}
abstract class Bird extends Animal {
   void sing() {
      System.out.print("chirp ");
   }
}
```

a) True
b) False

40. What does the following statements print to console?

Animal.java

```
public interface Animal {
   abstract void makeSound();
}
```

Main.java

```
class Dog implements Animal {
   @Override
   public void makeSound() {
         System.out.println("woof");
   }
}
class Cat implements Animal {
   @Override
   public void makeSound() {
         System.out.println("meow");
   }
}
public class Main {
   public static void main(String[] args) {
         Dog dog = new Dog();
         Cat cat = new Cat();
         Animal dogAnimal = dog;
         dog.makeSound();
         cat.makeSound();
         dogAnimal.makeSound();
   }
}
```

a) will not compile
b) meow meow meow
c) woof meow meow
d) woof meow woof
e) none of the above

41. True or False: The following will compile despite implementing an interface without implementing the abstract method.

Animal.java

```
public interface Animal {
   abstract void makeSound();
}
```

Main.java

```
class CookieMonster implements Animal {
   void eat() {
         System.out.print("nom nom nom ");
   }
}
```

a) True
b) False

42. Running the following piece of code

```
// print absolute path to file
System.out.println(file.getAbsolutePath());
// print current working directory
System.out.println(System.getProperty("user.dir"));
```

prints the following console:

```
C:\Users\usrName\workspace\ProjectName\folder\file.txt
C:\Users\usrName\workspace\ProjectName
```

Multiple true/false question: Which of the following works for declaring and initializing the file variable to the same file?

a) File file = new File("folder\file.txt");
b) File file = new File("folder\\file.txt");
c) File file = new File("C:\\Users\\usrName\\workspace\\ProjectName\\folder\\file.txt");
d) File file = new File("C:\\Users\\usrName\\workspace\\ProjectName\\.\\folder\\file.txt");
e) File file = new File("package\\..\\file.txt");

43. True or False: The "static" keyword means that the variable is constant and cannot be changed.

a) True
b) False

44. Placeholder