1. Which answer choice can replace line 6 so the code continues to produce the same output?

```
3:   List<String> rug = new ArrayList<>();
4:   rug.add("circle");
5:   rug.add("square");
6:   System.out.println(rug);
```

   A. `System.out.println(rug.asString);`

   B. `System.out.println(rug.asString());`

   C. `System.out.println(rug.toString);`

   D. `System.out.println(rug.toString());`

2. Which best describes this code?

```
class Stats {
   private int data;
   public int getData() {
      return data;
   }
   public void setData(int data) {
     this.data = data;
   }
}
```

   A. It is a singleton.

   B. It is well encapsulated.

   C. It is immutable.

   D. It is both well encapsulated and immutable.

3. What design pattern or principle ensures that there will be no more than one instance of a class?

   A. Encapsulation

   B. Immutability

   C. Singleton

   D. Static

4. What is the output of this code?

```
class Laptop extends Computer {
    public void startup() {
        System.out.print("laptop-");
    }
}
public class Computer {
   public void startup() {
      System.out.print("computer-");
   }
   public static void main(String[] args) {
      Computer computer = new Laptop();
      Laptop laptop = new Laptop();
      computer.startup();
      laptop.startup();
   }
}
```

A. `computer-laptop-`

B. `laptop-computer-`

C. `laptop-laptop-`

D. None of the above

5. Which method can be correctly inserted into this class to meet the contract of the `equals()` method? You may assume that `text` is not `null`.

```
class Button {
   private String text;

   public int hashCode() {
      return text.hashCode();
   }
}
```

A.

```
public boolean equals(Object o) {
if ( o == null ) return true;
if (! (o instanceof Button)) return false;
return text.equals(o.text);
}
```

B.

```
public boolean equals(Object o) {
```

```
if ( o == null ) return true;
Button b = (Button) o;
return text.equals(b.text);
}
```

C.

```
public boolean equals(Object o) {
if (! (o instanceof Button)) return false;
return text.equals(o.text);
}
```

D.

```
public boolean equals(Object o) {
if (! (o instanceof Button)) return false;
Button b = (Button) o;
return text.equals(b.text);
}
```

6. Fill in the blanks: ____________means the state of an object cannot be changed while ____________means that it can.
   A. Immutability, mutability
   B. Rigidity, flexibility
   C. Static, instance
   D. None of the above
7. Which is the first line to fail to compile?

```
class Tool {
   void use() { }     // r1
}

class Hammer extends Tool {
   private void use() { }  // r2
   public void bang() { }  // r3
}
```

   A. `r1`
   B. `r2`

C. `r3`

D. None of the above

8. Which of these classes properly implement(s) the singleton pattern?

```
class ExamAnswers {
  private static ExamAnswers instance = new ExamAnswers();
  private List<String> answers = new ArrayList<>();
  public static List<String> getAnswers() {
    return instance.answers;
  }
}
class TestAnswers {
   private static TestAnswers instance = new TestAnswers();
   private List<String> answers = new ArrayList<>();
   public static TestAnswers getTestAnswers() {
      return instance;
   }
   public List<String> getAnswers() {
      return answers;
   }
}
```

A. `ExamAnswers`

B. `TestAnswers`

C. Both classes

D. Neither class

9. What does the following print?

```
public class Transport {
   static interface Vehicle {}
   static class Bus implements Vehicle {}

   public static void main(String[] args) {
      Bus bus = new Bus();
      boolean n = null instanceof Bus;
      boolean v = bus instanceof Vehicle;
      boolean b = bus instanceof Bus;
      System.out.println(n + " " + v + " " + b);
  }
}
```

A. `true true true`

B. `false true true`

C. `false false false`

D. None of the above

10. What technique allows multiple variables from the same class to be shared across all instances of a class?

    A. Encapsulation

    B. Immutability

    C. Singleton

    D. Static

11. Which is not a requirement for a class to be immutable?

    A. A private constructor is provided.

    B. Any instance variables are private.

    C. Methods cannot be overridden.

    D. There are no setter methods.

12. Which statement is true about encapsulation while providing the broadest access allowed?

    A. Variables are `public` and methods are `private`.

    B. Variables are `public` and methods are `public`.

    C. Variables are `private` and methods are `public`.

    D. Variables are `private` and methods are `private`.

13. What does the following print?

```
class Laptop extends Computer {
   String type = "laptop";
}
public class Computer {
   String type = "computer";
   public static void main(String[] args) {
      Computer computer = new Laptop();
      Laptop laptop = new Laptop();
      System.out.print(computer.type + "," + laptop.type);
   }
}
```

    A. `computer,laptop`

B. `laptop,computer`

C. `laptop,laptop`

D. None of the above

14. Which of these classes is/are immutable?

```
public final class Flower {
   private final String name;
   private final List<Integer> counts;
   public Flower(String name, List<Integer> counts) {
      this.name = name;
      this.counts = counts;
   }
   public String getName() {
      return name;
   }
   public List<Integer> getCounts() {
      return counts;
   }
}

public final class Plant {
   private final String name;
   private final List<Integer> counts;
   public Plant(String name, List<Integer> counts) {
      this.name = name;
      this.counts = new ArrayList<>(counts);
   }
   public String getName() {
      return name;
   }
   public List<Integer> getCounts() {
      return new ArrayList<>(counts);
   }
}
```

A. `Flower`

B. `Plant`

C. Both classes

D. Neither class

15. Which methods compile?

```
private static int numShovels;
private int numRakes;

public int getNumShovels() {
   return numShovels;
}

public int getNumRakes() {
   return numRakes;
}
```

A. Just `getNumRakes()`

B. Just `getNumShovels()`

C. Both methods

D. Neither method

16. Which methods compile?

```
private static int numShovels;
private int numRakes;

public static int getNumShovels() {
   return numShovels;
}

public static int getNumRakes() {
   return numRakes;
}
```

A. Just `getNumRakes()`

B. Just `getNumShovels()`

C. Both methods

D. Neither method

17. How many lines of the main method fail to compile?

```
11:  static interface Vehicle {}
12:  static class Bus implements Vehicle {}
13:
14:  public static void main(String[] args) {
15:     Bus bus = new Bus();
16:
17:     System.out.println(null instanceof Bus);
18:     System.out.println(bus instanceof Vehicle);
19:     System.out.println(bus instanceof Bus);
20:     System.out.println(bus instanceof ArrayList);
```

```
        System.out.println(bus instanceof Collection);
}
```

A. One

B. Two

C. Three

D. Four

18. Which variable declaration is the first line not to compile?

```
class Building {}
class House extends Building{}

public void convert() {
   Building b =  new Building();
   House h = new House();
   Building bh = new House();
   Building p = (House) b;
   House q = (Building) h;
   Building r = (Building) bh;
   House s = (House) bh;
}
```

A. `p`

B. `q`

C. `r`

D. `s`

19. Which statement is true about the code that can fill in the blank?

```
class Sticker {
   public int hashCode() {
      return 1;
   }
   public boolean equals(Object o) {
      return____________ ;
   }
}
```

A. It must return `false`.

B. It must return `true`.

C. It can return either `true` or `false`.

D. None of the above.

20. What change is needed to make `Secret` well encapsulated?

```
import java.util.*;

public class Secret {

   private int number = new Random().nextInt(10);
   public boolean guess(int candidate) {
      return number == candidate;
   }
}
```

A. Change `number` to use a `public` access modifier.

B. Declare a `private` constructor.

C. Remove the `guess` method.

D. None. It is already well encapsulated.

21. Which of these classes best implement(s) the singleton pattern?

```
class ExamAnswers {
   private static ExamAnswers instance = new ExamAnswers();
   private List<String> answers = new ArrayList<>();
   private ExamAnswers() {}
   public ExamAnswers getExamAnswers() {
      return instance;
   }
   public List<String> getAnswers() {
      return answers;
   }
}
class TestAnswers {
   private static TestAnswers instance = new TestAnswers();
   private List<String> answers = new ArrayList<>();
   private TestAnswers() {}
   public static TestAnswers getTestAnswers() {
      return instance;
   }
   public List<String> getAnswers() {
      return answers;
   }
}
```

A. `ExamAnswers`

B. `TestAnswers`

C. Both classes

D. Neither class

22. How many lines does the following code output?

```
public class Cars {
  static {
     System.out.println("static");
  }
  private static void drive() {
     System.out.println("fast");
  }
  public static void main(String[] args) {
     drive();
     drive();
  }
}
```

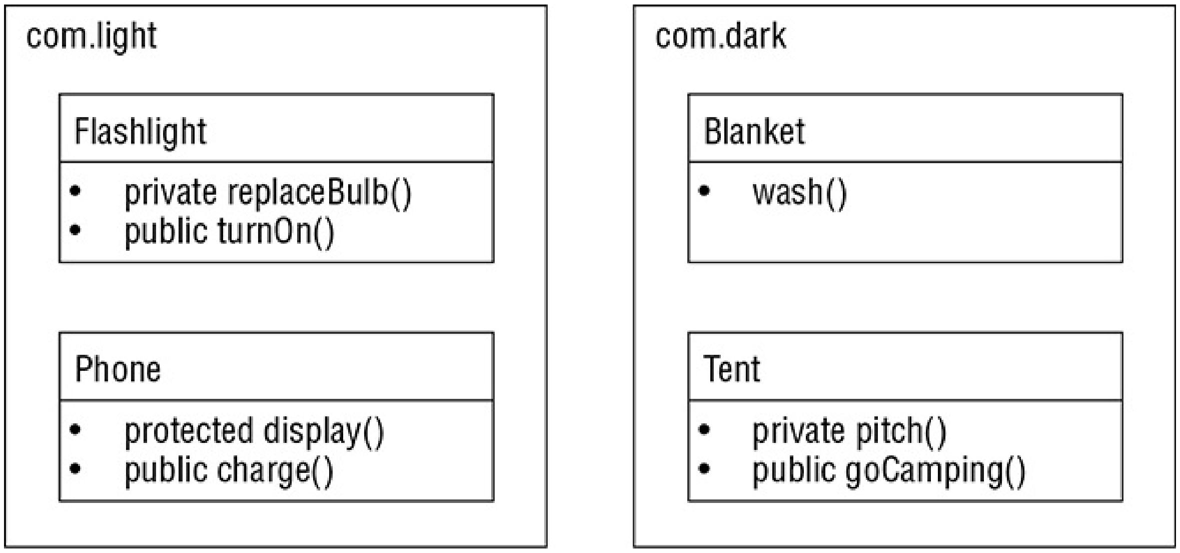
A. One

B. Two

C. Three

D. None of the above. The code does not compile.

23. Which is not a true statement given this diagram?



A. Instance methods in the `Blanket` class can call the `Flashlight` class's `turnOn()`.

B. Instance methods in the `Flashlight` class can call the `Flashlight` class's `replaceBulb()`.

C. Instance methods in the `Phone` class can call the `Blanket` class's `wash()`.

D. Instance methods in the `Tent` class can call the `Tent` class's `pitch()`.

24. Given the diagram in the previous question, how many of the classes can call the `display()` method?

A. One

B. Two

C. Three

D. Four

25. What does the following print?

```
class SmartWatch extends Watch {
   private String getType() { return "smart watch"; }
   public String getName(String suffix) {
      return getType() + suffix;
   }
}
public class Watch {
   private String getType() { return "watch"; }
   public String getName(String suffix) {
      return getType() + suffix;
   }
   public static void main(String[] args) {
      Watch watch = new Watch();
      SmartWatch smartWatch = new SmartWatch();
      System.out.print(watch.getName(","));
      System.out.print(smartWatch.getName(""));
   }
}
```

A. `smart watch,watch`

B. `watch,smart watch`

C. `watch,watch`

D. None of the above

26. What does the following print?

```
public class Transport {
   static interface Vehicle {}
   static class Bus implements Vehicle {}
   static class Van extends Bus {}

   public static void main(String[] args) {
      Bus bus = new Van();
      Van van = new Van();
      Van[] vans = new Van[0];

      boolean b = bus instanceof Vehicle;
      boolean v = van instanceof Vehicle;
      boolean a = vans instanceof Vehicle[];

      System.out.println(b + " " + v + " " + a);
   }
}
```

A. `true true true`

B. `false true true`

C. `true false false`

D. None of the above. The code does not compile

27. Which of the following correctly fills in the blank so this code compiles and prints `true`?

```
public class Button {
   private String text;
   public int hashCode() {
      return text.hashCode();
   }
   public boolean equals(Object o) {
      if (____________)  return false;
      Button b = (Button) o;
      return text.equals(b.text);
   }
   public static void main(String[] args) {
      Button b1 = new Button();
      Button b2 = new Button();
      b1.text = "mickey";
      b2.text = "mickey";
      System.out.println(b1.equals(b2));
   }
}
```

A. `(o instanceof Button)`

B. `(o instanceOf Button)`

C. `!(o instanceof Button)`

D. `!(o instanceOf Button)`

28. Which is the first line to fail to compile?

```
class Tool {
   void use() { }     // r1
}

class Hammer extends Tool {
   private void use(String s) { }  // r2
   public void bang() { }  // r3
}
```

A. `r1`

B. `r2`

C. `r3`

D. None of the above

29. What is lazy instantiation?

A. A technique that can be used in an immutable class to wait until the first use to create the object

B. A technique that can be used in a singleton to wait until the first use to create the object

C. A technique that can be used in an immutable class to save memory when creating the object

D. A technique that can be used in a singleton to save memory when creating the object

30. Which variable declaration is the first line not to compile?

```
30:  class Building {}
31:  class House extends Building{}
32:
33:  public void convert() {
34:     Building b =  new Building();
35:     House h = new House();
36:     Building bh = new House();
37:     House p = (House) b;
```

```
     House q = (House) h;
     House r = (House) bh;
 }
```

A. `p`

B. `q`

C. `r`

D. None of the above

31. Which statement about encapsulation is not true?

    A. Encapsulation allows putting extra logic in the getter and setter methods.

    B. Encapsulation can use immutable instance variables in the implementation.

    C. Encapsulation causes two classes to be more tightly tied together.

    D. Encapsulation makes it easier to change the instance variables in the future.

32. Which of these classes is/are immutable?

```
public class Flower {
   private final String name;
   private final List<Integer> counts;
   public Flower(String name, List<Integer> counts) {
      this.name = name;
      this.counts = new ArrayList<>(counts);
   }
   public final String getName() {
      return name;
   }
   public final List<Integer> getCounts() {
      return new ArrayList<>(counts);
   }
}

public class Plant {
   private final String name;
   private final List<Integer> counts;
   public Plant(String name, List<Integer> counts) {
      this.name = name;
      this.counts = new ArrayList<>(counts);
   }
   public String getName() {
      return name;
   }
   public List<Integer> getCounts() {
      return new ArrayList<>(counts);
   }
}
```

A. `Flower`

B. `Plant`

C. Both classes

D. Neither class

33. How many lines does the following code output?

```
public class Cars {
   private static void drive() {
       static {
          System.out.println("static");
       }
       System.out.println("fast");
   }
   public static void main(String[] args) {
      drive();
      drive();
   }
}
```

A. One

B. Two

C. Three

D. None of the above. The code does not compile.

34. How many of the following pairs of values can fill in the blanks to comply with the contract of the `hashCode()` and `equals()` methods?

```
class Sticker {
   public int hashCode() {
      return _____________;
   }
   public boolean equals(Object o) {
      return _____________;
   }
}
```

I. `1, false`

II. `1, true`

III. `new Random().nextInt(), false`

IV. `new Random().nextInt(), true`

A. None

B. One

C. Two

D. Three

35. How do you change the value of an instance variable in an immutable class?

A. Call the setter method.

B. Remove the final modifier and set the instance variable directly.

C. Use a method other than Option A or B.

D. You can't.

36. Which technique or pattern requires instance variables to implement?

A. Is-a

B. Object composition

C. Singleton

D. None of the above

37. How many lines of output does the following generate?

```
public class HowMany {
   static {
      System.out.println("any");
   }
   {
      System.out.println("more");
   }
   public static void main(String[] args) {
      new HowMany();
      new HowMany();
   }
}
```

A. Two

B. Three

C. Four

D. None of the above. The code does not compile.

38. Which is the first line to fail to compile?

```
class Tool {
   default void use() { }     // r1
}

class Hammer extends Tool {
   public void use() { }  // r2
   public void bang() { }  // r3
}
```

A. `r1`

B. `r2`

C. `r3`

D. None of the above

39. Which variable declaration is the first line to throw a `ClassCastException` at runtime?

```
class Building {}
class House extends Building{}

public void convert() {
   Building b =  new Building();
   House h = new House();
   Building bh = new House();
   House p = (House) b;
   House q = (House) h;
   House r = (House) bh;
}
```

A. `p`

B. `q`

C. `r`

D. None of the above

40. Which of the following values can fill in the blank for the class to be correctly implemented?

```
class Sticker {
   public int hashCode(Object o) {
      return____________ ;
   }
   public boolean equals(Object o) {
      return true;
   }
}
```

I. `-1`

II. `5`

III. `new Random().nextInt()`

A. I

B. I and II

C. I, II, and III

D. I and III