ANSWER KEY

1. **B**
2. **B**
3. **C**
4. **C**
5. **A**
6. **D**
7. **C**
8. **C**
9. **C**
10. **D**
11. **A**
12. **D**
13. **E**
14. **A**
15. **C**
16. **E**
17. **C**
18. **C**
19. **B**
20. **A**
21. **E**

ANSWERS EXPLAINED

1. **(B)** All the `Math` class methods are static methods, which means you can't use a `Math` object that calls the method. The method is invoked using the class name, `Math`, followed by the dot operator. Thus segment II is correct, and segment I is incorrect. Segment III will cause an error: Since the parameters of `pow` are of type `double`, the result should be stored in a `double`.

2. **(B)** The `Math.sqrt` method must be invoked on a primitive type `double`, but auto-unboxing takes care of that in the line

   ```
   double x = d;
   ```

   The return type of the method is `Double`, and autoboxing takes care of that in the statement

   ```
   return x;
   ```

   Segment III fails because you can't use the `Double` constructor to create a new object without using the keyword `new`.

3. **(C)** The value −4.67 must be rounded to −5. Subtracting 0.5 gives a value of −5.17. Casting to `int` truncates the number (chops off the decimal part) and leaves a value of −5. None of the other choices produces −5. Choice A gives the absolute value of `d`: 4.67. Choice B is an incorrect use of `Random`. The parameter for `nextInt` should be an integer n, $n \geq 2$. The method then returns a random `int` k, where $0 \leq k < n$. Choice D is the way to round a *positive* real number to the nearest integer. In the actual case it produces −4. Choice E gives the absolute value of −5, namely 5.

4. **(C)** The statement `double x = Math.random();` generates a random `double` in the range $0 \leq x < 1$. Suppose `probDeath` is 0.67, or 67%. Assuming that random doubles are uniformly distributed in the interval, one can expect that 67% of the time `x` will be in the range $0 \leq x < 0.67$. You can therefore simulate the probability of death by testing if x is between 0 and 0.67, that is, if $x < 0.67$. Thus, `x < probDeath` is the desired condition for plant death, eliminating choices A and B. Choices D and E fail because `(int) probDeath` truncates `probDeath` to 0. The test `x < 0` will always be false, and the test `x == 0` will only be true if the random number generator returned exactly 0, an extremely unlikely occurrence! Neither of these choices correctly simulates the probability of death.