› Answers and Explanations

Bullets mark each step in the process of arriving at the correct solution.

1. The answer is C.
   - A class is a blueprint for constructing objects of the type described by the class definition.
2. The answer is E.
   - The general form of a constructor declaration is:

     ```
     public NameOfClass(optional parameter list)
     ```

   - Notice that there is no return type, not even void.
   - In this example, the name of the class is `Cube`. III is a correct declaration for a no-argument constructor and IV is a correct declaration for a constructor with one parameter.
3. The answer is C.
   - The general form of an instantiation statement is:

     ```
     Type variableName = new Type(parameter list);
     ```

   - In this example, the type is `Student` and the variable name is `sam`.
   - We have to look at the constructor code to see that the parameter list consists of a `String` followed by a double.
   - The only answer that has the correct form with the parameters in the correct order is:

     ```
     Student sam = new Student("Sam Smith", 3.5);
     ```

4. The answer is B.
   - We do not have direct access to the instance variables of an object, because they are private. We have to ask the object for the data we need with an accessor method. In this case, we need to call the `getName` method.
   - We need to ask the specific object we are interested in for its name. The general syntax for doing that is:

     ```
     variableName.methodName();
     ```

   - Placing the call in the print statement gives us:

     ```
     System.out.println(sam.getName());
     ```

5. The answer is E.
   - A modifier or setter method has to take a new value for the variable from the caller and assign that value to the variable. The new value is given to the method through the use of a parameter.
   - A modifier method does not return anything. Why would it? The caller has given the value to the method. The method doesn't have to give it back.
   - So we need a void method that has one parameter. It has to change the value of gpa to the value of the parameter. Option E shows the only method that does that.
6. The answer is C.
   - The instantiation statements give us 2 objects: s1 and s2.
   - s1 references (points to) a Student object with the state information: name = "Brody Kai", gpa = 3.9
   - s2 references (points to) a Student object with the state information: name = "Charlie Cole", gpa = 3.2
   - When we execute the statement s2 = s1, s2 changes its reference. It now references the exact same object that s1 references, with the state information: name = "Brody Kai", gpa = 3.9
   - s2.setGpa(4.0); changes the gpa in the object that both s1 and s2 reference, so when s1.getGpa(); is executed, that's the value it retrieves and that's the value that is printed.

7. The answer is D.

- When the method call `calculate(var, count)` is executed, the arguments 9 and 2 are passed by value. That is, their values are copied into the parameters a and b, so a = 9 and b = 2 at the beginning of the method.
- a = 9 − 2 = 7
- int c = 2
- b = 7 * 7 = 49
- a = 49 − (7 + 2) = 40 and that is what will be returned.

8. The answer is B.

- When the method call `calculate(a, b)` is executed, the arguments 9 and 2 are passed by value, that is their values are copied into the parameters a and b. The parameters a and b are completely different variables from the a and b in the calling method and any changes made to a and b inside the method do not affect a and b in the calling method in any way.
- We know from problem #1 that calculate(9, 2) will return 40.
- 9 2 40 will be printed.

9. The answer is C.

- All instance variables in the classes we write must be declared private. In fact, the AP Computer Science A exam readers may deduct points if you do not declare your variables private. However, some built-in Java classes provide us with useful constants by making their fields public.
- We access these fields by using the name of the class followed by a dot, followed by the name of the field. We can tell that it is a constant because its name is written entirely in capital letters. Some other public constants that we use are Integer.MAX_VALUE and Integer.MIN_VALUE.
- Note that we can tell this is not a method call, because the name is not followed by (). The Math.pow(radius, 2) is a method call to a static method of the `Math` class.

10. The answer is D.

- Option A is incorrect. It correctly states that each object has its own variable space, and each object's variables will be different. However, it is not correct about it being impossible.
- Option B is incorrect. It says to use a constant, which doesn't make sense because she wants to be able to update its value as objects are created.
- Option C is incorrect. There is no one mutator method call that will access the instance variables of each object of the class.
- Option D is correct. A `static` variable or class variable is a variable held by the class and accessible to every object of the class.
- Option E is simply not true.

11. The answer is B.

- Method overloading allows us to use the same method name with different parameters.

12. The answer is E.

- The intent of the mystery method is to find and return the smallest of the three integers. We know that all possible conditions are covered by the if statements and one of them has to be true, but the compiler does not know that. The compiler just sees that all of the `return` statements are in if clauses, and so it thinks that if all the conditions are false, the method will not return a value. The compiler generates an error message telling us that the method must return a value.

13. The answer is A.

- The first time a `return` statement is reached, the method returns to the caller. No further code in the method will be executed, so no further returns will be made.

14. The answer is E.

- There are a lot of calls here, but only two that will affect the value of `val`, since the only time val is changed is in the constructor.
- It's important to notice that `val` is a static variable. That means that the class maintains one copy of the variable, rather than the objects each having their own copy. Since there is only one shared copy, changes made to `val` by any object will be seen by all objects.
- The first time the constructor is called is when the word "object" is instantiated. val = 31 when the constructor is called and this call to the constructor changes its value to 15 (integer division).
- The second time the constructor is called is when the sentence object is instantiated. This time val = 15 when the constructor is called and this call to the constructor changes its value to 7 (integer division).

15. The answer is D.

- Options I and V are incorrect. An overloaded method must have a different parameter list so that the compiler knows which version to use.
- Options II, III, and IV are valid declarations for overloaded methods. They can all be distinguished by their parameter lists.

16. On the AP Computer Science A exam, you are often asked to write an entire class, like you were in this question. Check your answer against our version below. The comments are to help you read our solution. Your solution does not need comments.

> **Make Variable Names Meaningful**
>
> It's OK if you use different variable names from what we use in the solutions to the chapter questions. Just be sure that the names are meaningful.

```
public class DreamVacation
{
   // Here are the private instance variables (parts a and b).
   private String destination;
   private double cost;

   // Here is the no-argument constructor (part c).
   public DreamVacation()
   {
   }

   // Here is the parameterized constructor (part d).
   public DreamVacation(String myDestination, double myCost)
   {
      destination = myDestination;
      cost = myCost;
   }

   // Here are the two accessor methods (part e).
   // Their names contain the word "get" since it is standard practice.
   public String getDestination()
```

```
    {
        return destination;
    }

    public double getCost()
    {
        return cost;
    }

    // Here are the two modifier methods (part f).
    // Their names contain the word "set" since it is standard practice.
    public void setDestination(String myDestination)
    {
        destination = myDestination;
    }

    public void setCost(double myCost)
    {
        cost = myCost;
    }
}
```

17. The `Height` class

```
public class Height
{
    // Part a - remember to simplify when these variables are changed
    private int feet;
    private int inches;

    // Part b - overloaded constructors
    // notice the calls to simplify.
    public Height(int i)
    {
        inches = i;
        simplify();
    }

    public Height(int f, int i)
    {
        feet = f;
        inches = i;
        simplify();
    }

    // Part c - This method puts feet and inches in simplest form
    // see note below for an alternate solution
    public void simplify()
    {
        int totalInches = 12 * feet + inches;
        feet = totalInches / 12;
        inches = totalInches % 12;
    }

    // Parts d and e - overloaded add methods
    // Notice the call to the Height object len passed as a parameter
    public void add(Height ht)
    {
        feet += ht.getFeet();
        inches += ht.getInches();
        simplify();
```

```
    }
    public void add(int in)
    {
        inches += in;
        simplify();
    }

    //Part f - getters for the instance variables inches and feet
    public int getInches()
    {
        return inches;
    }
    public int getFeet()
    {
        return feet;
    }
}
```

Note: There are many ways to write the `simplify` method. If you aren't sure if yours works, type it into your IDE. Here's one alternate solution:

```
while (inches > 12)
{
    inches -= 12;
    feet++;
}
```