chap 5: Writing Classes

1. The `Car` class will contain two string attributes for a car's make and model. The class will also contain a constructor.

```
public class Car
{
     /* missing code */
}
```

Which of the following replacements for `/* missing code */` is the most appropriate implementation of the class?

(A)
```
public String make;
public String model;
public Car(String myMake, String myModel)
{ /* implementation not shown */ }
```

(B)
```
public String make;
public String model;
private Car(String myMake, String myModel)
{ /* implementation not shown */ }
```

(C)
```
private String make;
private String model;
public Car(String myMake, String myModel)
{ /* implementation not shown */ }
```

(D)
```
public String make;
private String model;
private Car(String myMake, String myModel)
( /* implementation not shown */ }
```

(E)
```
private String make;
private String model;
private Car(String myMake, String myModel)
{ /* implementation not shown */ }
```

2. The `Date` class below will contain three `int` attributes for day, month, and year, a constructor, and a `setDate` method. The `setDate` method is intended to be accessed outside the class.

```
public class Date
{
     /* missing code */
}
```

Which of the following replacements for `/* missing code */` is the most appropriate implementation of the class?

chap 5: Writing Classes

```
    private int day;
    private int month;
    private int year;
(A) private Date()
    { /* implementation not shown */ }
    private void setDate(int d, int m, int y)
    { /* implementation not shown */ }

    private int day;
    private int month;
    private int year;
(B) public Date()
    { /* implementation not shown */ }
    private void setDate(int d, int m, int y)
    { /* implementation not shown */ }

    private int day;
    private int month;
    private int year;
(C) public Date()
    { /* implementation not shown */ }
    public void setDate(int d, int m, int y)
    { /* implementation not shown */ }

    public int day;
    public int month;
    public int year;
(D) private Date()
    { /* implementation not shown */ }
    private void setDate(int d, int m, int y)
    { /* implementation not shown */ }

    public int day;
    public int month;
    public int year;
(E) public Date()
    { /* implementation not shown */ }
    public void setDate(int d, int m, int y)
    { /* implementation not shown */ }
```

3. The `Player` class below will contain two `int` attributes and a constructor. The class will also contain a method `getScore` that can be accessed from outside the class.

```
public class Player
{
     /* missing code */
}
```

Which of the following replacements for `/* missing code */` is the most appropriate implementation of the class?

chap 5: Writing Classes

(A)
```
private int score;
private int id;
private Player(int playerScore, int playerID)
{ /* implementation not shown */ }
private int getScore()
{ /* implementation not shown */ }
```

(B)
```
private int score;
private int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
private int getScore()
{ /* implementation not shown */ }
```

(C)
```
private int score;
private int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
public int getScore()
{ /* implementation not shown */ }
```

(D)
```
public int score;
public int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
private int getScore()
{ /* implementation not shown */ }
```

(E)
```
public int score;
public int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
public int getScore()
{ /* implementation not shown */ }
```

chap 5: Writing Classes

4. Consider the following class declarations.

```
public class Alpha
{
     private int answer()
     {
         return 10;
     }
}

public class Beta
{
     public double sample()
     {
         Alpha item = new Alpha();
         double temp = item.answer();
         return temp * 2.0;
     }
}
```

Which of the following best describes why an error occurs when the classes are compiled?

(A) The class `Alpha` does not have a defined constructor.

(B) The class `Alpha` must be declared as a subclass of `Beta`.

(C) The class `Beta` must be declared as a subclass of `Alpha`.

(D) The `answer` method cannot be accessed from a class other than `Alpha`.

(E) The result of the method call `item.answer()` cannot be assigned to a variable of type `double`.

5. A bear is an animal and a zoo contains many animals, including bears. Three classes Animal, Bear, and Zoo are declared to represent animal, bear, and zoo objects. Which of the following is the most appropriate set of declarations?

chap 5: Writing Classes

(A)
```
public class Animal extends Bear
{
  ...
}
public class Zoo
{
  private Animal[] myAnimals;
  ...
}
```

(B)
```
public class Bear extends Animal
{
  ...
}
public class Zoo
{
  private Animal[] myAnimals;
  ...
}
```

(C)
```
public class Animal extends Zoo
{
  private Bear myBear;
  ...
}
```

(D)
```
public class Bear extends Animal, Zoo
{
  ...
}
```

(E)
```
public class Bear extends Animal implements Zoo
{
  ...
}
```

chap 5: Writing Classes

6. Consider the following class definition.

```
public class Element
{
     public static int max_value = 0;
     private int value;
     public Element (int v)
     {
         value = v;
         if (value > max_value)
         {
            max_value = value;
         }
     }
}
```

The following code segment appears in a class other than `Element`.

```
for (int i = 0; i < 5; i++)
{
     int k = (int) (Math.random() * 10 + 1);
     if (k >= Element.max_value)
     {
         Element e = new Element(k);
     }
}
```

Which of the following best describes the behavior of the code segment?

(A) Exactly 5 `Element` objects are created.

(B) Exactly 10 `Element` objects are created.

(C) Between 0 and 5 `Element` objects are created, and `Element.max_value` is increased only for the first object created.

(D) Between 1 and 5 `Element` objects are created, and `Element.max_value` is increased for every object created.

(E) Between 1 and 5 `Element` objects are created, and `Element.max_value` is increased for at least one object created.

chap 5: Writing Classes

7. Consider the following class definition.

```
public class WordClass
{
     private final String word;
     private static String max_word = "";
     public WordClass (String s)
     {
         word = s;
         if (word.length() > max_word.length())
         {
            max_word = word;
         }
     }
}
```

Which of the following is a true statement about the behavior of `WordClass` objects?

(A) A `WordClass` object can change the value of the variable `word` more than once.

(B) Every time a `WordClass` object is created, the `max_word` variable is referenced.

(C) Every time a `WordClass` object is created, the value of the `max_word` variable changes.

(D) No two `WordClass` objects can have their `word` length equal to the length of `max_word`.

(E) The value of the `max_word` variable cannot be changed once it has been initialized.

chap 5: Writing Classes

8. The following question is based on the following incomplete declaration of the class BoundedIntArray **and its constructor definitions.**

A BoundedintArray represents an indexed list of integers. In a BoundedIntArray the user can specify a size, in which case the indices range from 0 to size - 1. The user can also specify the lowest index, low, in which case the indices can range from low to low + size - 1.

```
public class BoundedIntArray
{
  private int[] myItems;   // storage for the list
  private int myLowIndex;  // lowest index

  public BoundedIntArray(int size)
  {
    myItems = new int[size];
    myLowIndex = 0;
  }

  public BoundedIntArray(int size, int low)
  {
    myItems = new int[size];
    myLowIndex = low;
  }
  // other methods not shown
}
```

Consider the following statements.

```
BoundedIntArray arrl = new BoundedIntArray(100, 5);
```

chap 5: Writing Classes

```
BoundedIntArray arr2 = new BoundedIntArray(100);
```

Which of the following best describes arrl and arr2 after these statements?

(A) arrl and arr2 both represent lists of integers indexed from 0 to 99.

(B) arrl and arr2 both represent lists of integers indexed from 5 to 104.

(C) arrl represents a list of integers indexed from 0 to 104, and arr2 represents a list of integers indexed from 0 to 99.

(D) arrl represents a list of integers indexed from 5 to 99, and arr2 represents a list of integers indexed from 0 to 99.

(E) arrl represents a list of integers indexed from 5 to 104, and arr2 represents a list of integers indexed from 0 to 99.

chap 5: Writing Classes

9. The following question is based on the following incomplete declaration of the class BoundedIntArray and its constructor definitions.

A BoundedintArray represents an indexed list of integers. In a BoundedIntArray the user can specify a size, in which case the indices range from 0 to size - 1. The user can also specify the lowest index, low, in which case the indices can range from low to low + size - 1.

```
public class BoundedIntArray
{
  private int[] myItems;   // storage for the list
  private int myLowIndex;  // lowest index

  public BoundedIntArray(int size)
  {
    myItems = new int[size];
    myLowIndex = 0;
  }

  public BoundedIntArray(int size, int low)
  {
    myItems = new int[size];
    myLowIndex = low;
  }
  // other methods not shown
}
```

Which of the following is the best reason for declaring the data fields myItems and myLowIndex to be private rather than public?

chap 5: Writing Classes

(A) This permits BoundedIntArray objects to be initialized and modified.

(B) This permits BoundedIntArray methods to be written and tested before code that uses a BoundedIntArray is written.

(C) This helps to prevent clients of the BoundedIntArray class from writing code that would need to be modified if the implementation of BoundedIntArray were changed.

(D) This prevents compile-time errors whenever public methods are called that access the private data fields.

(E) This prevents run-time errors whenever public methods are called that access the private data fields.

10. Consider the following class definition.

```
public class Box
{
     private double weight;

     /** Postcondition: weight is initialized to w. */
     public Box(double w)
     {
         /* implementation not shown */
     }

     public double getWeight()
     {
         return weight;
     }

     public void addWeight(double aw)
     {
         /* missing statement */
     }
}
```

The following code segment, which appears in a class other than `Box`, is intended to create a `Box` object `b1` with a weight of 2.2 units and then increase the weight of `b1` by 1.5 units.

```
Box b1 = new Box(2.2);
b1.addWeight(1.5);
```

Which of the following statements could replace `/* missing statement */` so that the code segment works as intended?

(A) `aw += weight;`

(B) `aw += getWeight();`

(C) `weight += aw;`

(D) `weight += getWeight();`

(E) `return weight + aw;`

chap 5: Writing Classes

11. Consider the following class definition.

```
public class RentalCar
{
     private double dailyRate;      // the fee per rental day
     private double mileageRate;    // the fee per mile driven
     public RentalCar(double daily, double mileage)
     {
         dailyRate = daily;
         mileageRate = mileage;
     }
     public double calculateFee(int days, int miles)
     {
         /* missing code */
     }
}
```

The `calculateFee` method is intended to calculate the total fee for renting a car. The total fee is equal to the number of days of the rental, `days`, times the daily rental rate plus the number of miles driven, `miles`, times the per mile rate.

Which of the following code segments should replace `/* missing code */` so that the `calculateFee` method will work as intended?

(A) `return dailyRate + mileageRate;`

(B) `return (daily * dailyRate) + (mileage * mileageRate);`

(C) `return (daily * days) + (mileage * miles);`

(D) `return (days * dailyRate) + (miles * mileageRate);`

(E) `return (days + miles) * (dailyRate + mileageRate);`

chap 5: Writing Classes

12. Consider the following class declaration.

```
public class Sample
{
     private int a;
     private double b;

     public Sample(int x, double y)
     {
         a = x;
         b = y;
     }

     // No other constructors
}
```

The following method appears in a class other than `Sample`.

```
public static void test()
{
     Sample object = new /* missing constructor call */ ;
}
```

Which of the following could be used to replace `/* missing constructor call */` so that the method will compile without error?

(A) `Sample()`

(B) `Sample(int x = 10, double y = 6.2)`

(C) `Sample(int x, double y)`

(D) `Sample(10, 6.2)`

(E) `Sample(6.2, 10)`

13. A car dealership needs a program to store information about the cars for sale. For each car, they want to keep track of the following information: number of doors (2 or 4), whether the car has air conditioning, and its average number of miles per gallon. Which of the following is the best object-oriented program design?

(A) Use one class, Car, with three instance variables: int numDoors, boolean hasAir, and double milesPerGallon.

(B) Use four unrelated classes: Car, Doors, AirConditioning, and MilesPerGallon.

(C) Use a class Car with three subclasses: Doors, AirConditioning, and MilesPerGallon.

(D) Use a class Car, with a subclass Doors, with a subclass AirConditioning, with a subclass MilesPerGallon.

(E) Use three classes: Doors, AirConditioning, and MilesPerGallon, each with a subclass Car.

chap 5: Writing Classes

14. Consider the following two methods that appear within a single class.

```
public void changeIt(int[] list, int num)
{
  list = new int[5];
  num = 0;

  for (int x = 0; x < list.length; x++)
    list[x] = 0;
}

public void start()
{
  int[] nums = {1, 2, 3, 4, 5};
  int value = 6;

  changeIt(nums, value);

  for (int k = 0; k < nums.length; k++)
    System.out.print(nums[k] + " ");

  System.out.print(value);
}
```

What is printed as a result of the call `start()`?

chap 5: Writing Classes

(A) 0 0 0 0 0 0

(B) 0 0 0 0 0 6

(C) 1 2 3 4 5 6

(D) 1 2 3 4 5 0

(E) `changeIt` will throw an exception.

15. Consider the following class definitions.

```
public class ClassA
{
     public String getValue()
     {
         return "A";
     }
     public void showValue()
     {
         System.out.print(getValue());
     }
}
public class ClassB extends ClassA
{
     public String getValue()
     {
         return "B";
     }
}
```

The following code segment appears in a class other than `ClassA` or `ClassB`.

```
ClassA obj = new ClassB();
obj.showValue();
```

What, if anything, is printed when the code segment is executed?

(A) `A`

(B) `B`

(C) `AB`

(D) `BA`

(E) Nothing is printed because the code does not compile.

chap 5: Writing Classes

16. Consider the definition of the `Person` class below. The class uses the instance variable `adult` to indicate whether a person is an adult or not.

```
public class Person

{
     private String name;
     private int age;
     private boolean adult;
     public Person (String n, int a)
     {
         name = n;
         age = a;
         if (age >= 18)
         {
            adult = true;
         }
         else
         {
            adult = false;
         }
     }
}
```

Which of the following statements will create a `Person` object that represents an adult person?

(A) `Person p = new Person ("Homer", "adult");`

(B) `Person p = new Person ("Homer", 23);`

(C) `Person p = new Person ("Homer", "23");`

(D) `Person p = new Person ("Homer", true);`

(E) `Person p = new Person ("Homer", 17);`

chap 5: Writing Classes

17. Consider the following class definition. Each object of the class `Item` will store the item's name as `itemName`, the item's regular price, in dollars, as `regPrice`, and the discount that is applied to the regular price when the item is on sale as `discountPercent`. For example, a discount of 15% is stored in `discountPercent` as 0.15.

```
public class Item
{
     private String itemName;
     private double regPrice;
     private double discountPercent;
     public Item (String name, double price, double discount)
     {
         itemName = name;
         regPrice = price;
         discountPercent = discount;
     }
     public Item (String name, double price)
     {
         itemName = name;
         regPrice = price;
         discountPercent = 0.25;
     }
     /* Other methods not shown */
}
```

Which of the following code segments, found in a class other than `Item`, can be used to create an item with a regular price of $10 and a discount of 25% ?

I. `Item b = new Item("blanket", 10.0, 0.25);`
II. `Item b = new Item("blanket", 10.0);`
III. `Item b = new Item("blanket", 0.25, 10.0);`

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I, II, and III

18. Consider the following method.

```
public void changeIt(int[] arr, int index, int newValue)
{
     arr[index] += newValue;
}
```

Which of the following code segments, if located in a method in the same class as `changeIt`, will cause the array `myArray` to contain `{0, 5, 0, 0}` ?

chap 5: Writing Classes

(A)
```
int[] myArray = new int[4];
changeIt(myArray, 1, 5);
```

(B)
```
int[] myArray = new int[4];
changeIt(myArray, 2, 5);
```

(C)
```
int[] myArray = new int[4];
changeIt(myArray, 5, 1);
```

(D)
```
int[] myArray = new int[5];
changeIt(myArray, 1, 4);
```

(E)
```
int[] myArray = new int[5];
changeIt(myArray, 1, 5);
```

19. Consider the following class, which uses the instance variable `balance` to represent a bank account balance.

```
public class BankAccount
{
     private double balance;
     public double deposit(double amount)
     {
         /* missing code */
     }
}
```

The `deposit` method is intended to increase the account balance by the deposit amount and then return the updated balance. Which of the following code segments should replace `/* missing code */` so that the `deposit` method will work as intended?

(A)
```
amount = balance + amount;
return amount;
```

(B)
```
balance = amount;
return amount;
```

(C)
```
balance = amount;
return balance;
```

(D)
```
balance = balance + amount;
return amount;
```

(E)
```
balance = balance + amount;
return balance;
```

chap 5: Writing Classes

20. Consider the following class definition.

```
public class Password
{
     private String password;
     public Password (String pwd)
     {
         password = pwd;
     }
     public void reset(String new_pwd)
     {
         password = new_pwd;
     }
}
```

Consider the following code segment, which appears in a method in a class other than `Password`. The code segment does not compile.

```
Password p = new Password("password");
System.out.println("The new password is " + p.reset("password"));
```

Which of the following best identifies the reason the code segment does not compile?

(A) The code segment attempts to access the private variable `password` from outside the `Password` class.

(B) The new password cannot be the same as the old password.

(C) The `Password` class constructor is invoked incorrectly.

(D) The `reset` method cannot be called from outside the `Password` class.

(E) The `reset` method does not return a value that can be printed.

21. Consider the following class declaration.

```
public class Circle
{
     private double radius;
     public double computeArea()
     {
         private double pi = 3.14159;
         public double area = pi * radius * radius;
         return area;
     }
     // Constructor not shown.
}
```

Which of the following best explains why the `computeArea` method will cause a compilation error?

chap 5: Writing Classes

(A) Local variables declared inside a method cannot be declared as `public` or `private`.

(B) Local variables declared inside a method must all be `private`.

(C) Local variables declared inside a method must all be `public`.

(D) Local variables used inside a method must be declared at the end of the method.

(E) Local variables used inside a method must be declared before the method header.

22. Consider the following class definition.

```
public class Info
{
     private String name;
     private int number;
     public Info(String n, int num)
     {
         name = n;
         number = num;
     }
     public void changeName(String newName)
     {
         name = newName;
     }
     public int addNum(int n)
     {
         num += n;
         return num;
     }
}
```

Which of the following best explains why the class will not compile?

(A) The class is missing an accessor method.

(B) The instance variables `name` and `number` should be designated `public` instead of `private`.

(C) The return type for the `Info` constructor is missing.

(D) The variable `name` is not defined in the `changeName` method.

(E) The variable `num` is not defined in the `addNum` method.

chap 5: Writing Classes

23. Consider the following class definition.

```
public class ItemInventory
{
     private int numItems;
     public ItemInventory(int num)
     {
         numItems = num;
     }
     public updateItems(int newNum)
     {
         numItems = newNum;
     }
}
```

Which of the following best identifies the reason the class does not compile?

(A) The constructor header is missing a return type.

(B) The `updateItems` method is missing a return type.

(C) The constructor should not have a parameter.

(D) The `updateItems` method should not have a parameter.

(E) The instance variable `numItems` should be `public` instead of `private`.

24. Consider the following `Bugs` class, which is intended to simulate variations in a population of bugs. The population is stored in the method's `int` attribute. The `getPopulation` method is intended to allow methods in other classes to access a `Bugs` object's population value; however, it does not work as intended.

```
public class Bugs
{
     private int population;

     public Bugs(int p)
     {
         population = p;
     }

     public int getPopulation()
     {
         return p;
     }
}
```

Which of the following best explains why the `getPopulation` method does NOT work as intended?

chap 5: Writing Classes

(A) The `getPopulation` method should be declared as `private`.

(B) The return type of the `getPopulation` method should be `void`.

(C) The `getPopulation` method should have at least one parameter.

(D) The variable `population` is not declared inside the `getPopulation` method.

(E) The instance variable `population` should be returned instead of `p`, which is local to the constructor.

25. Consider the following class definition.

```
public class Example
{
     private int x;
     // Constructor not shown.
}
```

Which of the following is a correct header for a method of the `Example` class that would return the value of the `private` instance variable `x` so that it can be used in a class other than `Example` ?

(A) `private int getX()`

(B) `private void getX()`

(C) `public int getX()`

(D) `public void getX()`

(E) `public void getX(int x)`

chap 5: Writing Classes

26. Consider the following class definition.

```
public class FishTank
{
     private double numGallons;
     private boolean saltWater;
     public FishTank(double gals, boolean sw)
     {
         numGallons = gals;
         saltWater = sw;
     }
     public double getNumGallons()
     {
         return numGallons;
     }
     public boolean isSaltWater()
     {
         if (saltWater)
         {
            return "Salt Water";
         }
         else
         {
            return "Fresh Water";
         }
     }
}
```

Which of the following best explains the reason why the class will not compile?

(A) The variable `numGallons` is not declared in the `getNumGallons` method.

(B) The variable `saltWater` is not declared in the `isSaltWater` method.

(C) The `isSaltWater` method does not return the value of an instance variable.

(D) The value returned by the `getNumGallons` method is not compatible with the return type of the method.

(E) The value returned by the `isSaltWater` method is not compatible with the return type of the method.

chap 5: Writing Classes

27. Consider the following class definition. The class does not compile.

```
public class Player
{
     private double score;
     public getScore()
     {
         return score;
     }
     // Constructor not shown
}
```

The accessor method `getScore` is intended to return the score of a `Player` object. Which of the following best explains why the class does not compile?

(A) The `getScore` method should be declared as `private`.

(B) The `getScore` method requires a parameter.

(C) The return type of the `getScore` method needs to be defined as `double`.

(D) The return type of the `getScore` method needs to be defined as `String`.

(E) The return type of the `getScore` method needs to be defined as `void`.

chap 5: Writing Classes

28. Consider the following class definition.

```
public class SomeClass
{
     private int x = 0;
     private static int y = 0;
     public SomeClass(int pX)
     {
         x = pX;
         y++;
     }
     public void incrementY()
     { y++; }
     public void incrementY(int inc)
     { y += inc; }
     public int getY()
     { return y; }
}
```

The following code segment appears in a class other than `SomeClass`.

```
SomeClass first = new SomeClass(10);
SomeClass second = new SomeClass(20);
SomeClass third = new SomeClass(30);
first.incrementY();
second.incrementY(10);
System.out.println(third.getY());
```

What is printed as a result of executing the code segment if the code segment is the first use of a `SomeClass` object?

(A) `0`

(B) `1`

(C) `11`

(D) `14`

(E) `30`

chap 5: Writing Classes

29. Consider the following `while` loop. Assume that the `int` variable `k` has been properly declared and initialized.

```
while (k < 0)
{
      System.out.print("*");
      k++;
}
```

Which of the following ranges of initial values for `k` will guarantee that at least one `"*"` character is printed?

I. `k < 0`
II. `k = 0`
III. `k > 0`

(A) I only

(B) III only

(C) I and II only

(D) II and III only

(E) I, II, and III

chap 5: Writing Classes

30. Consider the following class declaration.

```
public class IntCell
{
  private int myStoredValue;
  // constructor not shown
  public int getValue()
  {
    return myStoredValue;
  }
  public String toString ()
  {
    return "" + myStoredValue;
  }
}
```

Assume that the following declaration appears in a client class.

```
IntCell m = new IntCell();
```

Which of these statements can be used in the client class?

I. `System.out.println(m.getValue());`

II. `System.out.println(m.myStoredValue);`

III. `System.out.println(m);`

(A) I only
(B) II only
(C) III only
(D) I and II
(E) I and III

chap 5: Writing Classes

31. Consider the following instance variables and method that appear in a class representing student information.

```
private int assignmentsCompleted;

private double testAverage;


public boolean isPassing()

{ /* implementation not shown */ }
```

A student can pass a programming course if at least one of the following conditions is met.

- The student has a test average that is greater than or equal to 90.
- The student has a test average that is greater than or equal to 75 and has at least 4 completed assignments.

Consider the following proposed implementations of the `isPassing` method.

I.
```
if (testAverage >= 90)
   return true;
if (testAverage >= 75 && assignmentsCompleted >= 4)
   return true;
return false;
```

II.
```
boolean pass = false;
if (testAverage >= 90)
   pass = true;
if (testAverage >= 75 && assignmentsCompleted >= 4)
   pass = true;
return pass;
```

III.
```
return (testAverage >= 90) ||
```

chap 5: Writing Classes

```
(testAverage >= 75 && assignmentsCompleted >= 4);
```

Which of the implementations will correctly implement method isPassing?

(A) I only
(B) II only
(C) I and III only
(D) II and III only
(E) I, II, and III

32. Consider the following class definitions.

```
public class Item
{
     private int ID;
     public Item (int id)
     {
         ID = id;
     }
     public int getID()
     {
         return ID;
     }
     public void addToCollection (ItemCollection c)
     {
         c.addItem(this);
     }
}
public class ItemCollection
{
     private int last_ID;
     public void addItem(Item i)
     {
         if (i.getID() == last_ID)
         {
            System.out.print("ID " + i.getID() + " rejected; ");
         }
         else
         {
            last_ID = i.getID();
            System.out.print("ID " + i.getID() + " accepted; ");
         }
     }
     // Constructor not shown.
}
```

Consider the following code segment, which appears in a class other than `Item` or `ItemCollection`.

```
Item i = new Item(23);
Item j = new Item(32);
ItemCollection c = new ItemCollection();
i.addToCollection(c);
j.addToCollection(c);
j.addToCollection(c);
i.addToCollection(c);
```

What is printed as a result of executing the code segment?

chap 5: Writing Classes

(A) `ID 23 accepted; ID 32 accepted; ID 32 rejected; ID 23 accepted;`
(B) `ID 23 accepted; ID 32 accepted; ID 32 rejected; ID 23 rejected;`
(C) `ID 23 accepted; ID 32 rejected; ID 32 rejected; ID 23 accepted;`
(D) `ID 23 accepted; ID 32 rejected; ID 32 rejected; ID 23 rejected;`
(E) `ID 23 accepted; ID 32 rejected; ID 32 accepted; ID 23 rejected;`

33. Consider the following two methods, which appear within a single class.

```
public static void changeIt(int[] arr, int val, String word)
{
  arr = new int[5];
  val = 0;
  word = word.substring(0, 5);

  for (int k = 0; k < arr.length; k++)
  {
    arr[k] = 0;
  }
}


public static void start()
{
  int[] nums = {1, 2, 3, 4, 5};
  int value = 6;
  String name = "blackboard";

  changeIt(nums, value, name);

  for (int k = 0; k < nums.length; k++)
  {
    System.out.print(nums[k] + " ");
  }

  System.out.print(value + " ");
  System.out.print(name);
}
```

What is printed as a result of the call start() ?

(A) 0 0 0 0 0 0 black
(B) 0 0 0 0 0 6 blackboard
(C) 1 2 3 4 5 6 black
(D) 1 2 3 4 5 0 black
(E) 1 2 3 4 5 6 blackboard

chap 5: Writing Classes

34.

```
public void mystery(int[] data)
{
  for (int k = 0; k < data.length - 1; k++)
    data[k + 1] = data[k] + data[k + 1];
}
```

Consider the following method.

The following code segment appears in another method in the same class.

```
int[] values = {5, 2, 1, 3, 8};
mystery(values);
for (int v : values)
  System.out.print(v + " ");
System.out.println();
```

What is printed as a result of executing the code segment?

(A) 5 2 1 3 8

(B) 5 7 3 4 11

(C) 5 7 8 11 19

(D) 7 3 4 11 8

(E) Nothing is printed because an ArrayIndexOutOfBoundsException is thrown during the execution of method mystery.

chap 5: Writing Classes

35. Consider the following declaration of the class NumSequence, which has a constructor that is intended to initialize the instance variable seq to an ArrayList of numberOfValues random floating-point values in the range [0.0, 1.0).

```
public class NumSequence
{
  private ArrayList<Double> seq;

  // precondition: numberOfValues > 0
  // postcondition: seq has been initialized to an ArrayList of
  //                length numberOfValues; each element of seq
  //                contains a random Double in the range [0.0, 1.0)
  public NumSequence(int numberOfValues)
  {
    /* missing code */
  }
}
```

Which of the following code segments could be used to replace /* *missing code* */ so that the constructor will work as intended?

I.
```
ArrayList<Double> seq = new ArrayList<Double>();
for (int k = 0; k < numberOfValues; k++)
  seq.add(new Double(Math.random()));
```

II.
```
seq = new ArrayList<Double>();
for (int k = 0; k < numberOfValues; k++)
```

chap 5: Writing Classes

```
    seq.add(new Double(Math.random()));
```

III.
```
ArrayList<Double> temp = new ArrayList<Double>();
for (int k = 0; k < numberOfValues; k++)
  temp.add(new Double(Math.random()));

seq = temp;
```

(A) II only
(B) III only
(C) I and II
(D) I and III
(E) II and III

chap 5: Writing Classes

36. The question refer to the following declarations.

```
public class Point
{
  private double myX;
  private double myyY;
  // postcondition: this Point has coordinates (0,0)
  public Point ()
  { /* implementation not shown */ }
  // postcondition: this Point has coordinates (x,y)
  public Point(double x, double y)
  { /* implementation not shown */ }
  // other methods not shown
}
public class Circle
{
  private Point myCenter;
  private double myRadius;
  // postcondition: this Circle has center at (0, 0) and radius 0.0
  public Circle()
  { /* implementation not shown */ }
  // postcondition: this Circle has the given center and radius
  public Circle(Point center, double radius)
  { /* implementation not shown */ }
  // other methods not shown
}
```

chap 5: Writing Classes

In a client program which of the following correctly declares and initializes Circle circ with center at (29.5, 33.0) and radius 10.0 ?

(A) Circle circ = new Circle(29.5, 33.0, 10.0);

(B) Circle circ = new Circle((29.5, 33.0), 10.0);

(C) Circle circ = new Circle(new Point (29.5, 33.0), 10.0);

(D)
```
Circle circ = new Circle();
circ.myCenter = new Point(29.5, 33.0);
circ.myRadius = 10.0;
```

(E)
```
Circle circ = new Circle();
circ.myCenter = new Point();
circ.myCenter.myX = 29.5;
circ.myCenter.myY = 33.0;
cire.myRadius = 10.0;
```

chap 5: Writing Classes

37. The question refer to the following declarations.

```
public class Point
{
  private double myX;
  private double myyY;
  // postcondition: this Point has coordinates (0,0)
  public Point ()
  { /* implementation not shown */ }
  // postcondition: this Point has coordinates (x,y)
  public Point(double x, double y)
  { /* implementation not shown */ }
  // other methods not shown
}
public class Circle
{
  private Point myCenter;
  private double myRadius;
  // postcondition: this Circle has center at (0, 0) and radius 0.0
  public Circle()
  { /* implementation not shown */ }
  // postcondition: this Circle has the given center and radius
  public Circle(Point center, double radius)
  { /* implementation not shown */ }
  // other methods not shown
}
```

chap 5: Writing Classes

Which of the following would be the best specification for a Circle method isInside that determines whether a Point lies inside this Circle?

(A) public boolean isInside()

(B) public void isInside(boolean found)

(C) public boolean isInside(Point p)

(D) public void isInside(Point p, boolean found)

(E) public boolean isInside(Point p, Point center, double radius)

chap 5: Writing Classes

38. Consider the following class declarations.

```
public class Point
{
  private double x;   // x-coordinate
  private double y;   // y-coordinate

  public Point()
  {
    x = 0;
    y = 0;
  }

  public Point(double a, double b)
  {
    x = a;
    y = b;
  }

  // There may be instance variables, constructors, and methods that are not shown.
}


public class Circle
{
  private Point center;
  private double radius;

  /** Constructs a circle where (a, b) is the center and r is the radius.
   */
  public Circle(double a, double b, double r)
  {
    /* missing code */
  }
}
```

Which of the following replacements for /* missing code */ will correctly implement the Circle constructor?

chap 5: Writing Classes

I.
```
center = new Point();
radius = r;
```

II.
```
center = new Point(a, b);
radius = r;
```

III.
```
center = new Point();
center.x = a;
center.y = b;
radius = r;
```

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

39. Consider the following class definition.

```
public class Points
{
     private double num1;
     private double num2;
     public Points(int n1, int n2)            // Line 6
     {
         num1 = n1;                            // Line 8
         num2 = n2;                            // Line 9
     }
     public void incrementPoints(int value)   // Line 12
     {
         n1 += value;                          // Line 14
         n2 += value;                          // Line 15
     }
}
```

The class does not compile. Which of the following identifies the error in the class definition?

(A) In line 6, the `Points` constructor must have a `void` return type.
(B) In lines 8 and 9, `int` values cannot be assigned to `double` variables.
(C) In line 12, the `incrementPoints` method must have a non-`void` return type.
(D) In lines 14 and 15, the variables `n1` and `n2` are not defined.
(E) In lines 14 and 15, the variable `value` is not defined.

chap 5: Writing Classes

40. The following method is intended to return a string containing the character at position `n` in the string `str`. For example, `getChar("ABCDE", 2)` should return `"C"`.

```
/* missing precondition */
public String getChar(String str, int n)
{
     return str.substring(n, n + 1);
}
```

Which of the following is the most appropriate precondition for the method so that it does not throw an exception?

(A) `/* Precondition: 0 < n < str.length() - 1 */`

(B) `/* Precondition: 0 <= n <= str.length() - 1 */`

(C) `/* Precondition: 0 <= n <= str.length() */`

(D) `/* Precondition: n > str.length() */`

(E) `/* Precondition: n >= str.length() */`

41. Consider the following method, which is intended to return the product of `3` and the nonnegative difference between its two `int` parameters.

```
public int threeTimesDiff (int num1, int num2)
{
     return 3 * (num1 - num2);
}
```

Which, if any, precondition is required so that the method works as intended for all values of the parameters that satisfy the precondition?

(A) `num1 > 0, num2 > 0`

(B) `num1 >= 0, num2 >= 0`

(C) `num1 >= num2`

(D) `num2 >= num1`

(E) No precondition is required.

chap 5: Writing Classes

42. In the `Toy` class below, the `raisePrice` method is intended to increase the value of the instance variable `price` by the value of the parameter `surcharge`. The method does not work as intended.

```
public class Toy
{
     private String name;
     private double price;
     public Toy(String n, double p)
     {
         name = n;
         price = p;
     }
     public void raisePrice(double surcharge) // Line 12
     {
         return price + surcharge; // Line 14
     }
```

Which of the following changes should be made so that the class definition compiles without error and the method `raisePrice` works as intended?

(A) Replace line 14 with `surcharge += price;`.

(B) Replace line 14 with `price += surcharge;`.

(C) Replace line 14 with `return price += surcharge;`.

(D) Replace line 12 with `public raisePrice (double surcharge).`

(E) Replace line 12 with `public double raisePrice (double surcharge).`

chap 5: Writing Classes

43. A rectangular box fits inside another rectangular box if and only if the height, width, and depth of the smaller box are each less than the corresponding values of the larger box. Consider the following three interface declarations that are intended to represent information necessary for rectangular boxes.

I.
```
public interface RBox
{
  /** @return the height of this RBox */
  double getHeight();

  /** @return the width of this RBox */
  double getWidth();

  /** @return the depth of this RBox */
  double getDepth();
}
```

II.
```
public interface RBox
{
  /** @return true if the height of this RBox is less than the height of other;
   *          false otherwise
   */
  boolean smallerHeight(RBox other);

  /** @return true if the width of this RBox is less than the width of other;
   *          false otherwise
   */
  boolean smallerWidth(RBox other);

  /** @return true if the depth of this RBox is less than the depth of other;
   *          false otherwise
   */
  boolean smallerDepth(RBox other);
}
```

III.
```
public interface RBox
{
  /** @return the surface area of this RBox */
  double getSurfaceArea();

  /** @return the volume of this RBox */
  double getVolume();
}
```

Which of the interfaces, if correctly implemented by a Box class, would be sufficient functionality for a user of the Box class to determine if one Box can fit inside another?

chap 5: Writing Classes

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) I, II, and III

44. Consider the following class declarations.

```
public class Shoe
{
     private String shoeBrand;
     private String shoeModel;

     public Shoe(String brand, String model)
     {
         shoeBrand = brand;
         shoeModel = model;
     }

     // No other constructors
}

public class Boot extends Shoe
{
     private double heelHeight;

     public Boot(String brand, String model, double height)
     {
         /* missing implementation */
     }
}
```

Which of the following should be used to replace `/* missing implementation */` so that all instance variables are initialized with parameters?

(A)
```
shoeBrand = brand;
shoeModel = model;
heelHeight = height;
```

(B)
```
super();
heelHeight = height;
```

(C)
```
super(brand, model);
```

(D)
```
heelHeight = height;
super(brand, model);
```

(E)
```
super(brand, model);
heelHeight = height;
```

chap 5: Writing Classes

Directions: Select the choice that best fits each statement. The following question(s) refer to the following information.

Consider the following partial class declaration.

```
public class SomeClass
{
  private int myA;
  private int myB;
  private int myC;

  // Constructor(s) not shown

  public int getA()
  {  return myA;  }

  public void setB(int value)
  {  myB = value;  }
}
```

45. Which of the following changes to SomeClass will allow other classes to access but not modify the value of myC ?

(A) Make myC public.

(B) Include the method:
```
public int getC()
{  return myC;  }
```

(C) Include the method:
```
private int getC()
{  return myC;  }
```

(D) Include the method:
```
public void getC(int x)
{  x = myC;  }
```

(E) Include the method:
```
private void getC(int x)
{  x = myC;  }
```

chap 5: Writing Classes

46. Consider the following class definition.

```
public class Something
{
     private static int count = 0;
     public Something()
     {
         count += 5;
     }
     public static void increment()
     {
         count++;
     }
}
```

The following code segment appears in a method in a class other than `Something`.

```
Something s = new Something();
Something.increment();
```

Which of the following best describes the behavior of the code segment?

(A) The code segment does not compile because the `increment` method should be called on an object of the class `Something`, not on the class itself.

(B) The code segment creates a `Something` object `s`. The class `Something`'s static variable `count` is initially `0`, then increased by `1`.

(C) The code segment creates a `Something` object `s`. The class `Something`'s static variable `count` is initially `0`, then increased by `5`, then increased by `1`.

(D) The code segment creates a `Something` object `s`. After executing the code segment, the object `s` has a `count` value of `1`.

(E) The code segment creates a `Something` object `s`. After executing the code segment, the object `s` has a `count` value of `5`.

chap 5: Writing Classes

47. Consider the following class that stores information about temperature readings on various dates.

```
public class TemperatureReading implements Comparable
{
private double temperature; private int month, day, year;
public int compareTo(Object obj)
{
TemperatureReading other = (TemperatureReading) obj;
/* missing code */
}
// There may be instance variables, constructors, and methods that are not shown.
}
```

Consider the following code segments that are potential replacements for /* *missing code* */.

I. Double d1 = new Double(temperature); Double d2 = new Double(other.temperature);
return d1.compareTo(d2);

II. if (temperature < other.temperature)
return -1;
else if (temperature == other.temperature)

```
        return 0;
    else
        return 1;
```

III. return (int) (temperature - other.temperature);

Which of the code segments could be used to replace /* *missing code* */ so that compareTo can be used to order TemperatureReading objects by increasing temperature value?

(A) II only
(B) I and II only
(C) I and III only
(D) II and III only
(E) I, II, and III

48. Consider the following methods.

```
public void changer(String x, int y)
{
  x = x + "peace";
  y = y * 2;
}

public void test()
{
  String s = "world";
  int n = 6;
  changer(s, n);

  /* End of method */
 }
```

When the call test ( ) is executed, what are the values of s and n at the point indicated by / * *End of method* * / ?

chap 5: Writing Classes

(A) s / n
world / 6

(B) s / n
worldpeace / 6

(C) s / n
world / 12

(D) s / n
worldpeace / 12

(E) s / n
peace / 12

49. Consider the following class definition.

```
public class Tester
{
     private int num1;
     private int num2;
     /* missing constructor */
}
```

The following statement appears in a method in a class other than `Tester`. It is intended to create a new `Tester` object `t` with its attributes set to `10` and `20`.

```
Tester t = new Tester(10, 20);
```

Which of the following can be used to replace `/* missing constructor */` so that the object `t` is correctly created?

chap 5: Writing Classes

(A)
```
public Tester(int first, int second)
{
     num1 = first;
     num2 = second;
}
```

(B)
```
public Tester(int first, int second)
{
     num1 = 1;
     num2 = 2;
}
```

(C)
```
public Tester(int first, int second)
{
     first = 1;
     second = 2;
}
```

(D)
```
public Tester(int first, int second)
{
     first = 10;
     second = 20;
}
```

(E)
```
public Tester(int first, int second)
{
     first = num1;
     second = num2;
}
```

chap 5: Writing Classes

Directions: Select the choice that best fits each statement. The following question(s) refer to the following incomplete class declaration.

```
public class TimeRecord
{
  private int hours;
  private int minutes; // 0 ≤ minutes < 60
  /** Constructs a TimeRecord object.
   *  @param h the number of hours
   *         Precondition: h ≥ 0
   *  @param m the number of minutes
   *         Precondition: 0 ≤ m < 60
   */
  public TimeRecord(int h, int m)
  {
    hours = h;
    minutes = m;
  }

  /** @return the number of hours
   */
  public int getHours()
  { /* implementation not shown */ }

  /** @return the number of minutes
   *  Postcondition: 0 ≤ minutes < 60
   */
  public int getMinutes()
  { /* implementation not shown */ }

  /** Adds h hours and m minutes to this TimeRecord.
   *  @param h the number of hours
   *         Precondition: h ≥ 0
   *  @param m the number of minutes
   *         Precondition: m ≥ 0
   */
  public void advance(int h, int m)
  {
    hours = hours + h;
    minutes = minutes + m;
    /* missing code */
  }
  // Other methods not shown
}
```

50. Which of the following can be used to replace / * *missing code* * / so that advance will correctly update the time?

(A) minutes = minutes % 60;

(B) minutes = minutes + hours % 60;

(C) hours = hours + minutes / 60;
minutes = minutes % 60;

(D) hours = hours + minutes % 60;
minutes = minutes / 60;

(E) hours = hours + minutes / 60;

chap 5: Writing Classes

51. Consider the following instance variables and incomplete method that are part of a class that represents an item. The variables `years` and `months` are used to represent the age of the item, and the value for `months` is always between 0 and 11, inclusive. Method `updateAge` is used to update these variables based on the parameter `extraMonths` that represents the number of months to be added to the age.

```
private int years;

private int months; // 0 <= months <= 11


// precondition: extraMonths >= 0

public void updateAge(int extraMonths)

{

  /* body of updateAge */

}
```

Which of the following code segments could be used to replace `/* body of updateAge */` so that the method will work as intended?

I.
```
int yrs = extraMonths % 12;

int mos = extraMonths / 12;

years = years + yrs;

months = months + mos;
```

II.
```
int totalMonths = years * 12 + months + extraMonths;

years = totalMonths / 12;

months = totalMonths % 12;
```

III.
```
int totalMonths = months + extraMonths;

years = years + totalMonths / 12;

months = totalMonths % 12;
```

chap 5: Writing Classes

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) I, II, and III

52. Consider the following class definitions.

```
public class MenuItem
{
     private double price;
     public MenuItem(double p)
     {
         price = p;
     }
     public double getPrice()
     {
         return price;
     }
     public void makeItAMeal()
     {
         Combo meal = new Combo(this);
         price = meal.getComboPrice();
     }
}
public class Combo
{
     private double comboPrice;
     public Combo(MenuItem item)
     {
         comboPrice = item.getPrice() + 1.5;
     }
     public double getComboPrice()
     {
         return comboPrice;
     }
}
```

The following code segment appears in a class other than `MenuItem` or `Combo`.

```
MenuItem one = new MenuItem(5.0);
one.makeItAMeal();
System.out.println(one.getPrice());
```

What, if anything, is printed as a result of executing the code segment?

chap 5: Writing Classes

(A) `1.5`

(B) `5.0`

(C) `6.5`

(D) `8.0`

(E) Nothing is printed because the code will not compile.

chap 5: Writing Classes

53. Consider the following class definitions.

```
public class Class1
{
     private int val1;
     public Class1()
     {
         val1 = 1;
     }
     public void init ()
     {
         Class2 c2 = new Class2();
         c2.init(this, val1);
     }
     public void update(int x)
     {
         val1 -= x;
     }
     public int getVal()
     {
         return val1;
     }
}
public class Class2
{
     private int val2;
     public Class2()
     {
         val2 = 2;
     }
     public void init(Class1 c, int y)
     {
         c.update(val2 + y);
     }
}
```

The following code segment appears in a method in a class other than `Class1` or `Class2`.

```
Class1 c = new Class1();
c.init();
System.out.println(c.getVal());
```

What, if anything, is printed as a result of executing the code segment?

chap 5: Writing Classes

(A) `2`

(B) `1`

(C) `0`

(D) `-2`

(E) Nothing is printed because the code segment does not compile.

chap 5: Writing Classes

54. Consider the `BankAccount` class below.

```
public class BankAccount
{
     private final String ACCOUNT_NUMBER;
     private double balance;
     public BankAccount(String acctNumber, double beginningBalance)
     {
         ACCOUNT_NUMBER = acctNumber;
         balance = beginningBalance;
     }
     public boolean withdraw(double withdrawAmount)
     {
         /* missing code */
     }
}
```

The class contains the `withdraw` method, which is intended to update the instance variable `balance` under certain conditions and return a value indicating whether the withdrawal was successful. If subtracting `withdrawAmount` from `balance` would lead to a negative balance, `balance` is unchanged and the withdrawal is considered unsuccessful. Otherwise, `balance` is decreased by `withdrawAmount` and the withdrawal is considered successful.

Which of the following code segments can replace `/* missing code */` to ensure that the `withdraw` method works as intended?

I.

```
if (withdrawAmount > balance)
{
     return "Overdraft";
}
else
{
     balance -= withdrawAmount;
     return true;
}
```

II.

```
if (withdrawAmount > balance)
{
     return false;
}
else
{
     balance -= withdrawAmount;
     return balance;
}
```

chap 5: Writing Classes

III.

```
if (withdrawAmount > balance)
{
     return false;
}
else
{
     balance -= withdrawAmount;
     return true;
}
```

(A) I only

(B) II only

(C) III only

(D) I and II only

(E) I, II, and III

chap 5: Writing Classes

55. The class `Worker` is defined below. The class includes the method `getEarnings`, which is intended to return the total amount earned by the worker.

```
public class Worker
{
     private double hourlyRate;
     private double hoursWorked;
     private double earnings;
     public Worker(double rate, double hours)
     {
         hourlyRate = rate;
         hoursWorked = hours;
     }
     private void calculateEarnings()
     {
         double earnings = 0.0;
         earnings += hourlyRate * hoursWorked;
     }
     public double getEarnings()
     {
         calculateEarnings();
         return earnings;
     }
}
```

The following code segment appears in a method in a class other than `Worker`. The code segment is intended to print the value `800.0`, but instead prints a different value because of an error in the `Worker` class.

```
Worker bob = new Worker(20.0, 40.0);
System.out.println(bob.getEarnings());
```

Which of the following best explains why an incorrect value is printed?

(A) The `private` variables `hourlyRate` and `hoursWorked` are not properly initialized.

(B) The `private` variables `hourlyRate` and `hoursWorked` should have been declared `public`.

(C) The `private` method `calculateEarnings` should have been declared `public`.

(D) The variable `earnings` in the `calculateEarnings` method is a local variable.

(E) The variables `hourlyRate` and `hoursWorked` in the `calculateEarnings` method are local variables.