Multiple Choice and True/False - Questions 1 – 19

1. class
2. blueprint
3. instance of a class
4. field
5. new
6. accessor
7. mutator
8. stale
9. constructor
10. shadows
11. default constructor
12. their parameter lists
13. binding
14. both b and c

T/F

1. True
2. True
3. True
4. False
5. False

Find the Error - Question 1 – 5

1. Missing void before class
2. Add “int” to value2 and 3 like value1
3. Need parenthesis so it looks like “Rectangle box = new Rectangle();”
4. Two constructors, put y with the first one
5. Rename methods because they are the same

Algorithm Workbench - Question 1

public class Pet {  
 private String name;  
 private String animal;  
 private int age;  
  
 //Constructors for Pet  
 public Pet(String name, String animal, int age)   
 {  
 this.name = name;  
 this.animal = animal;  
 this.age = age;  
 }  
  
 //Methods  
 public void setName(String name)   
 { this.name = name; }  
  
 public void setAnimal(String animal)   
 { this.animal = animal; }  
  
 public void setAge(int age)   
 { this.age = age; }  
  
 public String getName()   
 { return name; }  
  
 public String getAnimal()   
 { return animal; }  
  
 public int getAge()   
 { return age; }  
}

Short Answer - Question 1 – 12

1. Classes specify an object's blueprints. Individual objects made from classes are called instances.   
  
  
2. Instances are like homes built from blueprints; classes are like blueprints.   
  
3. Mutator methods alter data, whereas accessor methods retrieve it.   
  
4. To guarantee encapsulation and manage access, fields are set to private.

5. Only students in that class have access to private fields.   
  
6. new generates class instances.   
  
7. Classes are found, loaded, compiled, and linked by the compiler.   
  
8. An object's constructor initializes it.   
  
9. Should there be no declared constructor, Java offers a default one.   
  
10. We refer to it as a default constructor.   
  
11. Methods with the same name but distinct specifications.   
  
12. Increases the code's readability and versatility.