# Unit 09 MCQ: Inheritance

**This quiz has 18 questions.**

1. Consider the following class declarations.

```
public class Dog {
   private String name;

   public Dog() {
      name = "NoName";
   }
}

public class Poodle extends Dog {
   private String size;

   public Poodle(String s) {
      size = s;
   }
}
```

The following statement appears in a method in another class.

```
Poodle myDog = new Poodle("toy");
```

Which of the following best describes the result of executing the statement?

Ⓐ The `Poodle` variable `myDog` is a reference to an instantiated `Poodle` object. The instance variable `size` is initialized to `"toy"`. The instance variable `name` is not assigned a value.

Ⓑ The `Poodle` variable `myDog` is a reference to an instantiated `Poodle` object. The instance variable `size` is initialized to `"toy"`. An implicit call to the no-argument `Dog` constructor is made, initializing the instance variable `name` to `"NoName"`.

Ⓒ The `Poodle` variable `myDog` is a reference to an instantiated `Poodle` object. The instance variable `size` is initialized to `"toy"`. An implicit call to the no-argument `Dog` constructor is made, initializing the instance variable name to `"toy"`.

Ⓓ A runtime error occurs because `super` is not used to call the no-argument `Dog` constructor.

Ⓔ A runtime error occurs because there is no one-argument `Dog` constructor.

2. Consider the following class declarations.

```
public class Publication {

   private String title;

   public Publication() {
      title = "Generic";
   }

   public Publication(String t) {
      title = t;
   }
}

public class Book extends Publication {

   public Book() {
      super();
   }

   public Book(String t) {
      super(t);
   }
}
```

The following code segment appears in a method in another class.

```
Book myBook =
  new Book("Adventure Story"); // Line 1
Book yourBook = new Book(); // Line 2
```

Which of the following best describes the result of executing the code segment?

Ⓐ The variable `myBook` is assigned a reference to a new `Book` object created using the one-argument `Book` constructor, which uses `super` to set `myBook`'s title attribute to `"Adventure Story"`. The variable `yourBook` is assigned a reference to a new `Book` object created using the no-argument `Book` constructor, which uses `super` to set `yourBook`'s title attribute to an empty string.

Ⓑ The variable `myBook` is assigned a reference to a new `Book` object created using the no-argument `Book` constructor, which uses super to set `myBook`'s title attribute to `"Generic"`. The variable `yourBook` is assigned a reference to a new `Book` object created using `super` to call to the `Publication` no-argument constructor to set `yourBook`'s title attribute to `"Generic"`.

Ⓒ The variable `myBook` is assigned a reference to a new `Book` object created using the one-argument `Book` constructor, which uses super to set `myBook`'s title attribute to `"Adventure Story"`. The variable `yourBook` is assigned a reference to a new `Book` object created using `super` to call to the `Publication` no-argument constructor to set `yourBook`'s title attribute to `"Generic"`.

Ⓓ A runtime error occurs in `line 1` because the one-argument `Publication` constructor cannot be called from the one-argument `Book` constructor.

Ⓔ A runtime error occurs in `line 2` because the no-argument `Publication` constructor cannot be called from the no-argument `Book` constructor.

# Unit 09 MCQ: Inheritance

3. Consider the following class declarations.

```
public class Tree {
   private String treeVariety;
   public Tree() {
      treeVariety = "Oak";
   }

   public Tree(String variety) {
      treeVariety = variety;
   }
}

public class DeciduousTree extends Tree {
   public DeciduousTree(String variety) {
      super();
   }
}

public class EvergreenTree extends Tree {
   public EvergreenTree(String variety) {
      super(variety);
   }
}
```

The following code segment appears in a method in another class.

```
DeciduousTree tree1 =
   new DeciduousTree("Maple");
EvergreenTree tree2 =
   new EvergreenTree("Fir");
```

Which of the following best describes the result of executing the code segment?

Ⓐ The variable `tree1` is assigned a reference to a new `DeciduousTree` object created using the `DeciduousTree` constructor, which uses `super` to set `tree1`'s `treeVariety` attribute to `"Maple"`. The variable `tree2` is assigned a reference to a new `EvergreenTree` object created using the `EvergreenTree` constructor, which uses `super` to set `tree2`'s `treeVariety` attribute to `"Fir"`.

Ⓑ The variable `tree1` is assigned a reference to a new `DeciduousTree` object created using the `DeciduousTree` constructor, which uses `super` to set `tree1`'s `treeVariety` attribute to `"Oak"`. The variable `tree2` is assigned a reference to a new `EvergreenTree` object created using the `EvergreenTree` constructor, which uses `super` to set `tree2`'s `treeVariety` attribute to `"Fir"`.

Ⓒ The variable `tree1` is assigned a reference to a new `DeciduousTree` object created using the `DeciduousTree` constructor, which uses `super` to set `tree1`'s `treeVariety` attribute to `"Oak"`. The variable `tree2` is assigned a reference to a new `EvergreenTree` object created using the `EvergreenTree` constructor, which uses `super` to set `tree2`'s `treeVariety` attribute to `"Oak"`.

Ⓓ The code segment does not compile because the `DeciduousTree` and `EvergreenTree` constructors should not take a parameter.

Ⓔ The code segment does not compile because the `DeciduousTree` and `EvergreenTree` constructors do not correctly call a `Tree` constructor.

4. Consider the following classes.

```
public class Bird {
   public void sing() {
      System.out.println("Cheep");
   }
}

public class Duck extends Bird {
   public void sing() {
      System.out.println("Quack");
   }
}

public class Chicken extends Bird {
   // No methods defined
}

public class Rooster extends Chicken
{
   public void sing() {
     System.out.println("Cockadoodle
doo");
   }
}
```

The following statement appears in a method in another class.

```
someBird.sing();
```

Under which of the following conditions will the statement compile and run without error?

I. When the variable `someBird` has been declared as type `Duck`

II. When the variable `someBird` has been declared as type `Chicken`

III. When the variable `someBird` has been declared as type `Rooster`

Ⓐ I only

Ⓑ III only

Ⓒ I and II only

Ⓓ I and III only

Ⓔ I, II, and III

# Unit 09 MCQ: Inheritance

5. Consider the following class declarations.

```
public class Person {

   public void laugh() {
      System.out.print("Hahaha");
   }
}

public class EvilPerson extends Person {

   public void laugh() {
      System.out.print("Mwahahaha");
   }
}

public class Henchman extends EvilPerson{
   // No methods defined
}
```

The following code segment appears in a method in another class.

```
alice.laugh();
```

Under which of the following conditions will the code segment print `"Mwahahaha"`?

I. when `alice` references an object of type `Person`

II. when `alice` references an object of type `EvilPerson`

III. when `alice` references an object of type `Henchman`

Ⓐ II only

Ⓑ I and II only

Ⓒ I and III only

Ⓓ II and III only

Ⓔ I, II, and III

6. Consider the following class declarations.

```
public class ParentClass {

   public void wheelsOnTheBus() {
     System.out.println("round and round");
   }
}

public class SubClass extends ParentClass {

   public void wheelsOnTheBus() {
     System.out.println("are flat");
   }
}

public class SubSubClass extends ParentClass{
   // No methods defined
}
```

The following code segment appears in a method in another class.

```
obj.wheelsOnTheBus();
```

Under which of the following conditions will the code segment print "are flat" ?

I. when `obj` references an object of type `ParentClass`

II. when `obj` references an object of type `SubClass`

III. when `obj` references an object of type `SubSubClass`

Ⓐ I only

Ⓑ II only

Ⓒ I and II only

Ⓓ II and III only

Ⓔ I, II, and III

# Unit 09 MCQ: Inheritance

7. Consider the following class declarations.

```
public class Range {

   private int lowValue;

   public Range(int low) {
      lowValue = low;
   }

   public String toString() {
     return "This range starts with "
            + lowValue;
   }
}

public class ClosedRange extends Range {

   private int highValue;

   public ClosedRange(int low, int high){
      super(low);
      highValue = high;
   }

   public String toString() {
      return super.toString()
             + " and ends with "
             + highValue;
   }
}
```

A code segment appearing in a method in another class is intended to produce the following output.

```
This range starts with 1 and ends with 10
```

Which of the following code segments will produce this output?

Ⓐ
```
Range r1 = new Range(1);
System.out.println(r1);
```

Ⓑ
```
Range r2 = new Range(1, 10);
System.out.println(r2);
```

Ⓒ
```
ClosedRange r3=new ClosedRange(1,10);
System.out.println(r3);
```

Ⓓ
```
ClosedRange r4=new ClosedRange(10,1);
System.out.println(r4);
```

Ⓔ
```
ClosedRange r5 = new ClosedRange(10);
System.out.println(r5);
```

8. Consider the following class declarations.

```
public class Hat {

   private String size;

   public Hat(String s) {
      size = s;
   }

   public String toString() {
      return "Size " + size + " hat";
   }
}

public class BallCap extends Hat {

   private String team;

   public BallCap(String mySize,
                  String myTeam) {
      super(mySize);
      team = myTeam;
   }

   public String toString() {
      return super.toString()
             + " with "
             + team + " logo";
   }
}
```

A code segment located in a different class is intended to produce the following output.

```
Size L hat with Denver logo
```

Which of the following code segments will produce this output?

Ⓐ
```
BallCap myHat=new BallCap("L","Denver");
System.out.println(myHat);
```

Ⓑ
```
BallCap myHat=new BallCap("Denver","L");
System.out.println(myHat);
```

Ⓒ
```
BallCap myHat=new BallCap("L");
myHat.team = "Denver";
System.out.println(myHat);
```

Ⓓ
```
Hat myHat=new Hat("L", "Denver");
System.out.println(myHat);
```

Ⓔ
```
Hat myHat=new Hat("L");
myHat.team = "Denver";
System.out.println(myHat);
```

# Unit 09 MCQ: Inheritance

9. Consider the following class declarations.

```
public class Parent {
   public void first() {
      System.out.print("P");
      second();
   }
   public void second() {
      System.out.print("Q");
   }
}

public class Child extends Parent {
   public void first() {
      super.first();
   }
   public void second() {
      super.second();
      System.out.print("R");
   }
}

public class Grandchild extends Child {
   public void first() {
      super.first();
      System.out.print("S");
   }
   public void second() {
      super.second();
      System.out.print("T");
   }
}
```

Which of the following code segments, if located in another class, will produce the output "PQRTS" ?

Ⓐ
```
Parent a = new Parent();
a.first();
```

Ⓑ
```
Child b = new Child();
b.first();
```

Ⓒ
```
Child c = new Child();
c.second();
```

Ⓓ
```
Grandchild d = new Grandchild();
d.first();
```

Ⓔ
```
Grandchild e = new Grandchild();
e.second();
```

10. Consider the following class declarations.

```
public class MultiTool {
   private int blade;
   private int screwdriver;
   public MultiTool(int b, int s) {
      blade = b;
      screwdriver = s;
   }
}

public class DeluxeMultiTool
             extends MultiTool {
   private boolean compass;
   public DeluxeMultiTool(int b, int s,
                          boolean c) {
      super(b, s);
      compass = c;
   }
   public String getCompass() {
      return compass + "";
   }
}
```

The following code segment appears in a method in another class.

```
ArrayList<MultiTool> toolList =
   new ArrayList<MultiTool>();
MultiTool tool1 =
   new DeluxeMultiTool(4, 2, false);// L2
DeluxeMultiTool tool2 =
    new DeluxeMultiTool(3, 1, true);// L3
toolList.add(tool1);                // L4
toolList.add(tool2);                // L5
for (MultiTool tool : toolList) {
  System.out.print(tool.getCompass());// L8
}
```

The code segment does not compile. Which of the following best explains the cause of the error?

Ⓐ Line `L2` causes a compile-time error because the variable `tool1` is declared as type `MultiTool` but references a `DeluxeMultiTool` object.

Ⓑ Line `L3` causes a compile-time error because the variable `tool2` is declared as type `MultiTool` and references a `DeluxeMultiTool` object.

Ⓒ In line `L4`, `tool2` cannot be added to the `ArrayList` because it references an object of `DeluxeMultiTool` object.

Ⓓ In line `L5`, `tool2` cannot be added to the `ArrayList` because it was declared to be of type `DeluxeMultiTool`.

Ⓔ Line `L8` causes a compile-time error because the `getCompass` method is not defined for objects of type `MultiTool`.

# Unit 09 MCQ: Inheritance

11. Consider the following class definitions.

```
public class Thing {
   /* implementation not shown */
}

public class MoreThing extends Thing {
   /* implementation not shown */
}
```

The following code segment appears in a class other than `Thing` or `MoreThing`.

```
Thing[] arr = new MoreThing[3]; // line 1
Thing t1 = new Thing();
Thing t2 = new MoreThing();     // line 3
MoreThing t3 = new MoreThing();
arr[0] = t1;                    // line 5
arr[1] = t2;                    // line 6
arr[2] = t3;                    // line 7
```

Which of the following best explains the error in the code segment?

Ⓐ Line 1 will cause an error because the type used to declare `arr` and the type used to instantiate `arr` are different.

Ⓑ Line 3 will cause an error because the type used to declare `t2` and the type used to instantiate `t2` are different.

Ⓒ Line 5 will cause an error because the types of `arr[0]` and `t1` are different.

Ⓓ Line 6 will cause an error because the types of `arr[1]` and `t2` are different.

Ⓔ Line 7 will cause an error because the types of `arr[2]` and `t3` are different.

12. Consider the following class definitions.

```
public class Appliance {

   private int id;
   private String brand;

   public Appliance(int aId,
                    String aBrand) {
      /* implementation not shown */
   }

   public String display() {
      /* implementation not shown */
   }
}

public class Refrigerator extends Appliance
{
   private int numOfDoors;

   public Refrigerator(int rId,
                       String rBrand,
                       int rNumOfDoors) {
      /* implementation not shown */
   }
}
```

The following code segment appears in a class other than `Appliance` or `Refrigerator`.

```
public static void
   displayFeatures(Refrigerator r)
   {
      System.out.println(r.display());// L3
   }

   Appliance a1 =
      new Refrigerator(456, "AllBrand",
                       2); // L6

   Refrigerator a2 =
      new Refrigerator(789, "Xtreme",
                       3); // L7

   displayFeatures(a1); // Line 8
   displayFeatures(a2); // Line 9
```

Which of the following best explains why the code segment will not compile?

Ⓐ Line `L3` causes a compile-time error because the `Refrigerator` class is missing the `display()` method.

Ⓑ Line `L6` causes a compile-time error because the variable `a1` references an object of the wrong type.

Ⓒ Line `L7` causes a compile-time error because the variable `a2` references an object of the wrong type.

Ⓓ Line `L8` causes a compile-time error because the parameter `a1` in the call `displayFeatures(a1)` has been declared as an `Appliance`.

Ⓔ Line `L9` causes a compile-time error because the parameter `a2` in the call `displayFeatures(a2)` has been declared as a `Refrigerator`.

# Unit 09 MCQ: Inheritance

13. Consider the following class definitions.

```
public class First {

   public void output1() {
      output2();
   }

   public void output2() {
      output3();
   }

   public void output3() {
      System.out.print("First");
   }
}

public class Second extends First {

   public void output() {
      output1();
      output2();
      output3();
   }
}

public class Third extends Second {

   public void output3() {
      System.out.print("Third");
   }
}
```

The following code segment appears in a class other than `First`, `Second`, or `Third`.

```
First sec = new Second(); // Line 1
Second thr = new Third(); // Line 2
sec.output(); // Line 3
thr.output(); // Line 4
```

Which of the following best explains why the code segment will not compile?

Ⓐ Line 3 causes a compile-time error because the variable `sec` needs to be declared as type `Second` to call `output()`.

Ⓑ Line 4 causes a compile-time error because the variable `thr` needs to be declared as type `Third` to call `output()`.

Ⓒ Line 3 causes a compile-time error because the `Second` class is missing the `output1` method.

Ⓓ Line 3 causes a compile-time error because the `Second` class is missing the `output2` method.

Ⓔ Line 4 causes a compile-time error because the `Third` class is missing the `output` method.

14. Consider the following class definitions.

```
public class Person {

   private String firstName;
   private String lastName;

   public Person(String pFirstName,
                 String pLastName) {
      firstName = pFirstName;
      lastName  = pLastName;
   }

   public void personInfo() {
      System.out.println("My name is "
         + firstName + " " + lastName);
   }
}

public class Teacher extends Person {

   private String school;
   private String subject;

   public Teacher(String tFN,
                  String tLN,
                  String tSchool,
                  String tSubject) {
      super(tFN, tLN);
      school  = tSchool;
      subject = tSubject;
   }

   public void teacherInfo() {
      personInfo();
      System.out.println("I teach "
         + subject + " at " + school);
   }
}
```

The following code segment appears in a class other than `Person` or `Teacher`.

```
Person teach =
    new Teacher("Henry", "Lowe",
                "PS 150", "English");
teach.teacherInfo();
```

Which of the following best explains why the code segment will not compile?

Ⓐ The call to `personInfo()` in the `teacherInfo` method should be `this.personInfo()`; because `personInfo` is a method in the `Person` class.

Ⓑ The `personInfo` method should be moved to the `Teacher` class because `personInfo` is a method only in the `Person` class.

Ⓒ The `teacherInfo` method should be moved to the `Person` class because `teacherInfo` is a method in the `Teacher` class.

Ⓓ The variable `teach` should be declared as a `Teacher` data type because `teacherInfo` is a method in the `Teacher` class.

Ⓔ The variable `teach` should be instantiated as a `Person` object because `teach` has been declared as type `Person`.

# Unit 09 MCQ: Inheritance

15. Consider the following class definitions.

```
public class Aclass {
   public void methodX() {
      System.out.print("Super X ");
      methodY();
   }
   public void methodY() {
      System.out.print("Super Y ");
      methodZ();
   }
   public void methodZ() (
      System.out.print("Super Z");
   }
}

public class Bclass extends Aclass {
   public void methodX() {
      super.methodX();
   }
   public void methodY() {
      System.out.print("Sub Y ");
      methodZ();
   }
}
```

The following code segment appears in a class other than `Aclass` or `Bclass`.

```
Aclass thing = new Bclass();
thing.methodX();
```

The code segment is intended to display the following.

```
Super X Super Y Super Z
```

Which of the following best explains why the code segment does not work as intended?

Ⓐ The variable `thing` should be declared as a `Bclass` data type because `thing` is instantiated as a `Bclass` object.

Ⓑ The variable `thing` should be instantiated as an `Aclass` object because `methodY` is overridden in `Bclass`.

Ⓒ The method `methodX` should be removed from the `Aclass` definition because `methodX` is overridden in `Bclass`.

Ⓓ The method `methodY` should be removed from the `Aclass` definition because `methodY` is overridden in `Bclass`.

Ⓔ The method `methodZ` should be overridden in the `Bclass` definition because `methodZ` appears only in `Aclass`.

16. Consider the following class definitions.

```
public class Vehicle {
   private int numOfWheels;
   public Vehicle(int nNumOfWheels) {
      numOfWheels = nNumOfWheels;
   }
   public String toString() {
      return "Number of Wheels: "
             + numOfWheels;
   }
}

public class Motorized extends Vehicle{
   private int maxSpeed;
   public Motorized(int nNumOfWheels,
                    int nMaxSpeed) {
      super(nNumOfWheels);
      maxSpeed = nMaxSpeed;
   }
   public String toString() {
      String s = super.toString()
                 + " Max Speed: ";
      if (maxSpeed <= 10) {
         s += "Slow";
      }else if (maxSpeed>10 &&
                maxSpeed<=100) {
         s += "Fast";
      } else {
         s += "Super Speedy";
      }
      return s;
   }
}
```

Which of the following code segments will display:
`Number of Wheels: 4 Max Speed: Fast`
when executed in a class other than `Vehicle` or `Motorized`?

Ⓐ
```
Vehicle obj = new Vehicle(55);
System.out.println(obj);
```

Ⓑ
```
Vehicle obj = new Vehicle(55);
System.out.println(toString(obj));
```

Ⓒ
```
Motorized obj = new Motorized(55);
System.out.println(obj);
```

Ⓓ
```
Vehicle obj = new Motorized(4, 55);
System.out.println(obj);
```

Ⓔ
```
Motorized obj = new Motorized(4, 55);
System.out.println(toString(obj));
```

# Unit 09 MCQ: Inheritance

17. Consider the following class definition.

```
public class Silly {

  private int var1;
  private String var2;

  public Silly(int v1, String v2) {
     var1 = v1;
     var2 = v2;
  }

  public boolean matches(Silly s) {
     if (s == null) {
        return false;
     }
     return (var1 == s.var1 &&
             var1 == var2.length() &&
             var2.length() ==
                s.var2.length());
  }
}
```

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

```
Silly s1 = new Silly(3, "abcd");
Silly s2 = new Silly(3, "abcd");
Silly s3 = new Silly(5, "vwxyz");
Silly s4 = new Silly(5, "aaaaa");
Silly s5 = new Silly(5, "efg");
```

Which of the following `Boolean` expressions will evaluate to `true`?

Ⓐ s1.matches(s2)

Ⓑ s2.matches(s3)

Ⓒ s3.matches(s4)

Ⓓ s4.matches(s5)

Ⓔ s5.matches(s1)

18. Consider the following class definition.

```
public class Time {

  private int hours;
  private int minutes;

  public Time(int h, int m) {
     hours = h;
     minutes = m;
  }

  /** Returns true if:
   *       hours*60+minutes
   *  of this object is equal to
   *       hours*60+minutes
   *  of object other; otherwise
   *  returns false
   */
  public boolean equals(Object other){
     //implementation not shown
  }
}
```

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

```
Time t1 = new Time(1, 10);
Time t2 = new Time(0, 70);
```

Which of the following statements will print `true`?

I. `System.out.println(t1 == t2);`

II. `System.out.println(t1.equals(t2));`

III. `System.out.println(equals(t1,t2);`

Ⓐ I only

Ⓑ II only

Ⓒ III only

Ⓓ I and II only

Ⓔ I and III only