Examen de Programación 3 - Enero 2019 - 2ª parte - Cuestiones

- *Tiempo:* 75 minutos
- Cada respuesta correcta puntúa 0.3125 puntos. Las respuestas incorrectas no puntúan.
- Contesta en los espacios previstos al efecto.

Dado el código correcto siguiente, señala la respuesta correcta (puede haber más de una).

```
class A {
    private int a;
    public A() { a=0; }
    public A(int x) { a=x; }
    public int getA() { return a; }
}

class B extends A {
    private int b;
    public B() { b=0; }
    public B(int y) { b=y; }
    public int getB() { return b; }
}
```

1. Al invocar alguno de los constructores de B...
   - ◯ B.B() invoca implícitamente a A.A()
   - ◯ B.B(int) invoca implícitamente a A.A(int)
   - ◯ B.B(int) invoca implícitamente a A.A()
   - ◯ B.B(int) invoca implícitamente a A.A() y luego a A.A(int)
2. Cuando ejecutamos '`new B(3)`'...
   - ◯ Primero se ejecuta B.B(int) y luego A.A(int)
   - ◯ Primero se ejecuta B.B(int) y luego A.A()
   - ◯ Primero se ejecuta A.A(int) y luego B.B(int)
   - ◯ Primero se ejecuta A.A() y luego B.B(int)
3. Dada esta asignación: '`A objeto = new B();`'
   - ◯ podemos ejecutar'`objeto.getA()`'
   - ◯ podemos ejecutar '`objeto.getB()`'
   - ◯ no podemos ejecutar '`objeto.getA()`'
   - ◯ no podemos ejecutar '`objeto.getB()`'
4. Si tenemos la asignación '`A objeto = new B(3);`', el resultado de llamar a '`objeto.getA()`' es
   - ◯ el método devuelve 0
   - ◯ el método devuelve 3
   - ◯ un error de ejecución
   - ◯ un error de compilación
5. Si tenemos la asignación '`B objeto = new B(3);`', el resultado de llamar a '`objeto.getA()`' es
   - ◯ el método devuelve 0
   - ◯ el método devuelve 3
   - ◯ un error de ejecución

◯ un error de compilación

6. Si tenemos las siguientes instrucciones '`B objeto = (B) new A(3); objeto.getA();`', el resultado es
   - ◯ La llamada a `getA()` devuelve 0
   - ◯ La llamada a `getA()` devuelve 3
   - ◯ un error de ejecución
   - ◯ un error de compilación

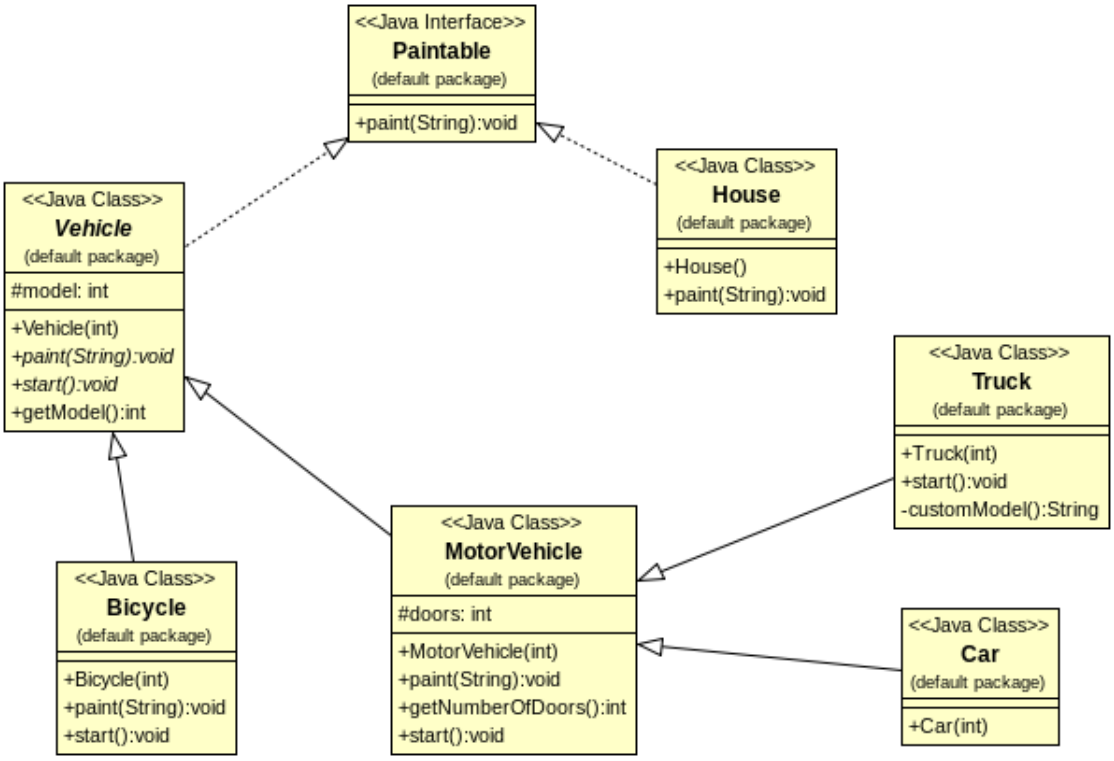

Figura 1: Diagrama de clases.

Las preguntas siguientes se refieren al diagrama de clases en la figura 1 y su código correcto correspondiente a continuación. Todas las clases están en el mismo paquete, incluída la clase `Main` que aparece en las preguntas.

```
public interface Paintable {
  void paint(String color);
}

public class House implements Paintable {
  @Override
  public void paint(String color) {
    System.out.println("House painted "+color);
  }
}

abstract public class Vehicle implements Paintable {
  protected int model;
  public Vehicle(int model) {
    super();
```

```
    this.model= model;
  }
  @Override
  abstract public void paint(String color);
  abstract public void start();
  public int getModel () {
    return model;
  }
}

public class Bicycle extends Vehicle {
  public Bicycle(int model) {
    super(model);
  }
  @Override
  public void paint(String color) {
    System.out.println("Bicycle "+model+" painted "+color);
  }
  @Override
  public void start() {
    System.out.println("Bicycle "+model+" moving");
  }
}

public class MotorVehicle extends Vehicle {
  protected int doors= 0;
  public MotorVehicle(int model) {
    super(model);
  }
  @Override
  public void paint(String color) {
    System.out.println("Motor vehicle "+model+" painted "+color);
  }
  public int getNumberOfDoors() {
    return doors;
  }
  @Override
  public void start() {
    System.out.println("Motor vehicle "+model+" moving");
  }
}

public class Truck extends MotorVehicle {
  public Truck(int model) {
    super(model);
    doors= 2;
  }
  @Override
  public void start () {
    System.out.println("Truck "+customModel()+" moving");
  }
  private String customModel () {
    return "T"+model;
  }
}

public class Car extends MotorVehicle {
  public Car(int model) {
    super(model);
    doors= 4;
  }
}
```

En los fragmentos de código de las preguntas, asumimos que existen las sentencias `import` adecuadas para usar las librerías estándar de Java (como `java.util`) y que pretendemos ejecutar el método '`main`'. Para cada una de las siguientes cuestiones indica una sola de estas tres posibles respuestas.

- "Error de compilación en la línea ___." y a continuación explica brevemente porqué se produce el error.
- "Error de ejecución en la línea ___." y a continuación explica brevemente porqué se produce el error.
- "Ejecuta correctamente" y a continuación indica la salida emitida por el programa tras ejecutar el método `main`.

En caso de que existan errores en distintas líneas del código, indica sólo el primero de ellos.

7. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3       List<Vehicle> list1 = new ArrayList<>();
4       List<Car> list2 = new ArrayList<>();
5       Car c = new Car(1000);
6       list1.add(c);
7       list2.add(c);
8       list1 = list2;
9      list1.get(0).start();
10   }
11 }
```

Respuesta:

8. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3         List<MotorVehicle> list1 = new ArrayList<>();
4         List<Truck> list2 = new ArrayList<>();
5       MotorVehicle v = new Truck(1000);
6       list1.add(v);
7       list2.add(v);
8       list1.get(0).start();
9       list2.get(0).start();  }
10 }
```

Respuesta:

9. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      Vehicle v= new Car(5000);
4      Car c= new Car(6000);
5      System.out.println(c.getNumberOfDoors());
6      System.out.println(v.getNumberOfDoors());
7    }
8  }
```

Respuesta:

10. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      MotorVehicle mv= new Truck(2000);
4      mv.start();
5    }
6  }
```

Respuesta:

11. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      Bicycle b= new Bicycle(2200);
4      b.start();
5      b.paint("yellow");
6      Vehicle v= new Vehicle();
7      v.start();
8      v.paint("red");
9    }
10 }
```

Respuesta:

12. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      Paintable p1= new Truck(2000);
4      Paintable p2= new Paintable();
5      p1.paint("white");
6      p2.paint("blue");
7    }
8  }
```

Respuesta:

13. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3        List<Paintable> list = new ArrayList<>();
4        list.add(new Truck(2000));
5      list.add(new House());
6           for (int i=0; i<list.size(); i++) {
7               list.get(i).paint("blue");
8           }
9    }
10 }
```

Respuesta:

14. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      Vehicle v1= new Car(3000);
4      MotorVehicle mv= (Car)v1;
5      mv.paint("blue");
6      Vehicle v2= new Bicycle(321);
7      Vehicle v3= (Bicycle)v2;
8      v3.paint("blue");
9    }
10 }
```

Respuesta:

15. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      Truck t= new Truck(2222);
4      Paintable p= (Paintable)(Vehicle)t;
5      p.paint("purple");
6      Paintable p2= t;
7      p2.paint("yellow");
8    }
9  }
```

Respuesta:

16. Código:

```
1  public class Main {
2    public static void main(String[] args) {
3      Paintable c= new Car(1111);
4      House h= (House)(Paintable)c;
5      h.paint("gray");
6    }
7  }
```

Respuesta: