Programming Language Concepts

Quiz 1, II Semester, 2024–2025

30 January, 2025

1. Consider the following Java code skeleton.

```
public abstract class Vehicle{
  public boolean equals(Vehicle a){ ... }
}

public class Bike extends Vehicle{
  public boolean equals(Bike b){ ... }
}

public class Car extends Vehicle{
  public boolean equals(Vehicle m){ ... }
}

public class TestVehicles{
  public static void main(String[] args){
    Bike meteor = new Bike();
    Car phantom = new Car();
    Object op = phantom;
    Vehicle vm = meteor;
    ...
  }
}
```

Against each of the following, tick the `equals()` method that is invoked, among `Object.equals()`, `Vehicle.equals()`, `Bike.equals()` and `Car.equals()`.

(a) `op.equals(phantom);`

`Object` ✔	`Vehicle`	`Bike`	`Car`

Explanation: The type of `op` is `Object`. The only `equals()` available for type `Object` is `Object.equals()`, so `op.equals(phantom)` invokes `Object.equals()`

(b) `vm.equals(meteor);`

`Object`	`Vehicle` ✔	`Bike`	`Car`

Explanation: The type of `vm` is `Vehicle`. The type of `meteor` is `Bike`. The two methods available to `vm` are `Object.equals(Object)` and `Vehicle.equals(Vehicle)`. The second one is the closest match, so `vm.equals(meteor)` invokes `Vehicle.equals()`.

(c) `meteor.equals(phantom);`

`Object`	`Vehicle` ✔	`Bike`	`Car`

Explanation: The type of `meteor` is `Bike`. The type of `phantom` is `Car`. The three methods available to `meteor` are `Object.equals(Object)`, `Vehicle.equals(Vehicle)` and `Bike.equals(Bike)`. The third option is not compatible with the argument `phantom`. Of the other two, the second one is the closest match, so `meteor.equals(phantom)` invokes `Vehicle.equals()`.

(d) `phantom.equals(meteor);`

`Object`	`Vehicle`	`Bike`	`Car` ✔

Explanation: The type of `phantom` is `Car`. The type of `meteor` is `Bike`. The three methods available to `phantom` are `Object.equals(Object)`, `Vehicle.equals(Vehicle)` and `Car.equals(Vehicle)`. The third option is the closest match, so `phantom.equals(meteor)` invokes `Car.equals()`.

2. Consider the following Java code fragment.

```
public class CreditCard {...}
public class ChipCard extends CreditCard{...
  // Adds a new instance variable chip_id of type int
  // and defines a new method readChip() that returns an int
}

ChipCard[] chips = new ChipCard[10];
CreditCard[] cards = chips;
```

For each of the following statements, select whether the statement is legal, generates a compile-time error, or generates a run-time error.

(a) `cards[4] = new CreditCard();`

Error	None	Compile	Runtime ✔

Explanation: `cards` has type `CreditCard[]`, so the compiler does not flag an error. At run time `cards` refers to an array of type `ChipCard[]`, so the assignment fails.

(b) `cards[7] = new ChipCard();`

Error	None ✔	Compile	Runtime

Explanation: `cards` has type `CreditCard[]` and `ChipCard` is a subtype of `CreditCard` so the compiler passes the code. At run time, `cards` has type `ChipCard[]`, so there is no error.

(c) `int v = chips[3].readChip();`

Error	None ✔	Compile	Runtime

Explanation: `chips` has type `ChipCard[]`, so `readChip()` is a valid method and there is no error.

(d) `int v = cards[8].readChip();`

Error	None	Compile ✔	Runtime

Explanation: `cards` has type `CreditCard[]` and `readChip()` is not defined for `CreditCard`, so this is flagged by the compiler.