Name:	Roll No:	

Programming Language Concepts

Quiz 1, II Semester, 2023–2024

30 January, 2024

1. Consider the following Java code skeleton.

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

public class Bird extends Animal{
  public boolean equals(Bird b) {...}
}

public class Mammal extends Animal{
  public boolean equals(Mammal m) {...}
}

public class TestAnimals{
  public static void main(String[] args){
    Bird ostrich = new Bird(...);
    Mammal dolphin = new Mammal(...);
    Object od = dolphin;
    Animal ao = ostrich;
}
```

Against each of the following, tick the `equals()` method that is invoked, among `Object.equals()`, `Animal.equals()`, `Bird.equals()` and `Mammal.equals()`.

(a) `od.equals(ostrich);`

Object ✔	Animal	Bird	Mammal

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

(b) `od.equals(dolphin);`

Object ✔	Animal	Bird	Mammal

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

(c) `ao.equals(ostrich);`

Object	Animal ✔	Bird	Mammal

Explanation: The type of `ao` is `Animal`. `ao.equals(ostrich)` looks for `equals()` that is compatible with `equals(Bird)`. The nearest match is `Animal.equals()`

(d) `ao.equals(dolphin);`

Object	Animal ✔	Bird	Mammal

Explanation: The type of `ao` is `Animal`. `ao.equals(dolphin)` looks for `equals()` that is compatible with `equals(Mammal)`. The nearest match is `Animal.equals()`

(e) `ostrich.equals(od);`

Object ✔	Animal	Bird	Mammal

Explanation: The type of `ostrich` is `Bird` but the type of `od` is `Object`. The only `equals()` available to `Bird` that is compatible with `equals(Object)` is `Object.equals()`

(f) `dolphin.equals(ao);`

Object	Animal ✔	Bird	Mammal

Explanation: The type of `dolphin` is `Mammal` and the type of `ao` is `Animal`. `Animal.equals()` is the nearest match for `equals(Animal)`.

2. Consider the following Java code fragment.

```
public class Ticket {...}
public class ETicket extends Ticket{...
  // Adds a new instance variable QRcode
  // and defines a new method getQRcode();
}

ETicket[] etktarr = new ETicket[10];
Ticket[] ticketarr = etktarr;
```

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

(a) `ticketarr[4] = new Ticket();` | *Error* | None | Compile | Runtime ✔

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

(b) `ticketarr[7] = new ETicket();` | *Error* | None ✔ | Compile | Runtime

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

(c) `QRcode q = etktarr[3].getQRcode();` | *Error* | None ✔ | Compile | Runtime

Explanation: `etktarr` has type `ETicket[]`, so `getQRcode()` is a valid method and there is no error.

(d) `QRcode q = ticketarr[8].getQRcode();` | *Error* | None | Compile ✔ | Runtime

Explanation: `ticketarr` has type `Ticket[]` and `getQRcode()` is not defined for `Ticket`, so this is flagged by the compiler.