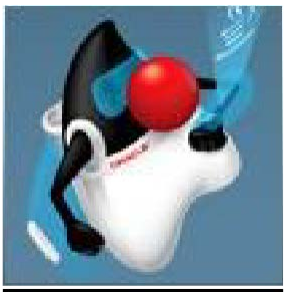

OCA 2: Object Orientation

Exercises

Q1 – JAT2Ex1

A Horse "HAS-A" Halter.

A Halter is a strap or a rope placed around the head of a horse or other animal, and is used for leading or tethering it.

A Horse object needs to be halted (stopped by pulling its halter). As halt behaviour is also characteristic of other animals such as Reindeer or Mule, it would be a poor design decision to include it only in the Horse class or to duplicate it across other animal classes.

Halter
- colour: String
+ halt(): String

Horse	**Reindeer**
+ Halter halter	+ Halter halter
+ halt(): String	+ halt(): String

The UML diagram shown above, shows the classes, which are to be modelled in this exercise.

Halter specific behaviour is stored exclusively in the Halter class. Any object which requires 'halter' behaviour (such as Horse and Reindeer) can store a reference to a Halter object and call its methods when required. Another programmer, who makes use of a Horse object does not need to be aware that the Horse object itself uses an external object (Halter), when performing halter-related behaviour.

You are now asked to model the solution.

Halter Class

Create a class named Halter.

The class should have one instance variable, a String named colour. The access modifier is private. The following constructors should be declared.

- A no-argument constructor, which calls an overloaded constructor, passing it the value "brown".
- A constructor that takes one argument, a String to represent the colour of the halter. The value of the instance variable should be set to the value passed to the constructor.

The class should contain one method named halt(). The access modifier is public. The method should return the following message, 'Halted….'.

Please Turn Over

Horse Class

Create a simple class to model a Horse.

The Horse class should contain one instance variable, a reference to a Halter object. The access modifier is private.

The Horse class should contain one method, named halt(), which should return a String. The method should return the name of the class ('Horse: ") concatenated with the String returned from the call to the halt() method of the Halter object (to halt the Horse).

Reindeer Class

Create another simple class to model a Reindeer.

The functionality of this class should be modelled as per the Horse class.

Main Class

Finally, create a class named Main, to contain the main method.

Within the main method(), create a Horse object. Call the Horse object's halt() method.

Create a Reindeer object. Call the Reindeer object's halt() method.

Save your classes in a folder named **JAT2Ex1**.

```
Horse: Halted...
Reindeer: Halted...
```

Finally:

Include a brief comment in the Main class to explain the difference between an "IS-A" and "HAS-A" class relationship in Java.

Please Turn Over

Q2 – JAT2Ex2

Whistle
+ blowWhistle(): String

Child	**Referee**
+ Whistle whistle	+ Whistle whistle
+ blowWhistle(): String	+ blowWhistle(): String

A Child object and a Referee object (a person who officiates a sporting event) both require access to a Whistle object. Create the following classes as shown in the above UML diagram.

Create a class named Main to contain the main method. Instantiate an object from the Child class and call its blowWhistle() method. Instantiate an object from the Referee class and call its blowWhistle() method.

The program output should appear similar to the screenshot below.

```
Child: Thweeeeeet!

Referee: Thweeeeeet!
```

Save your classes in a folder named **JAT2Ex2**.

Please turn over

Q3 - JAT2Ex3

Review the following classes and use a pen and paper to answer the questions that follow.

```
public interface Moveable{
  String move();
}
```

```
public class Animal implements Moveable{
   public String move(){
     return "Animal moving..";
   }

   public void sleep(int noOfHours){
     System.out.println("Animal sleeping for " + noOfHours + " hours.");
   }
}
```

```
public class Antelope extends Animal{
  @Override
  public String move(){
     return "Antelope moving..";
   }

  public void joinHerd(){
    System.out.println("Joining other Antelopes....");
   }

   public void sleep(int noOfHours, int noOfMinutes){
      System.out.println("Antelope sleeping for " + noOfHours + " hours and " + noOfMinutes + "
      minutes." );
   }
}
```

```
public class JAT2Ex3{
   public static void main (String[] args){

     Animal a = new Antelope();

     Object o = a;

     Moveable m = a;

     Antelope b = a;

     a.joinHerd();

     System.out.println(a.move());

     System.out.println(a.sleep(5,6));
  }
}
```

Please turn over

Q1: Animal a = new Antelope(); The code fails to compile. True or false.
Q2: Object o = a; The code fails to compile. True or false.
Q3: Moveable m = a; The code fails to compile. True or false.
Q4: Antelope b = a; The code fails to compile. True or false.
Q5: a.joinHerd(); The code fails to compile. True or false.
Q6: System.out.println(a.move()); The program will output, Animal moving. True or false.
Q7: System.out.println(a.sleep(5,6)); The program will output: Antelope sleeping for 5 hours and 6 minutes. True or false.

Are your answers correct? Create the program and find out.

Save your classes in a folder named **JAT2Ex3**.

Please turn over

Q4 - JAT2Ex4

Kids love a trip to the Zoo. The problem is that the animals don't perform their tricks when they are hungry.

It is the job of the Animal Keeper to feed the hungry animals.

In this question, you are asked to create a Java application, which simulates the act of feeding hungry animals at the zoo. You will be asked to create a number of classes.

Animal – (Abstract Class)

Create an abstract class named Animal. The class should have the following instance variables.

Access Modifier	Data Type	Name
private	String	name
private	int	age
private	boolean	isHungry

Create the following constructors:

- A constructor with three parameters (String, int, boolean).
- A constructor with no parameters. In this constructor, call the other (overloaded) constructor and pass to it the following values ("Unknown",0,false).

Create setter and getter methods for each of the instance variables.

An Animal cannot be assigned a negative age. If a negative age is specified by the user, it should be set to 0 (zero). Include this data validation in the setter method for the instance variable, *age* and in the constructor which accepts three parameters.

Behaviours

Animals are fed in different ways. Create an abstract method named feed(). The method should return a String.

Animals can perform various tricks. Create an abstract method named performTrick(). The method should return a String.

Subclasses

Create the following classes to model animals typically found at the Zoo. Each class should subclass the Animal abstract class and implement the abstract behaviours in a specific manner.

Elephant	Penguin
feed(): "Elephant <<insert name>> is being fed, he eats hay."	**feed()**:"Penguin <<insert name>> is being fed. He eats fish from a bucket."
performTrick(): "Elephant <<insert name>> blows with his trunk!"	**performTrick()**:"Penguin <<insert name>>flaps wings".

AnimalKeeper

Create a class to model an Animal Keeper.

The Animal Keeper has to **feed** many animals at the zoo.

In the AnimalKeeper class, use polymorphism to come up with an optimal design to allow the AnimalKeeper to **feed** any animal.

Please Turn Over

Main

Create a class to contain the main method.

- Create an AnimalKeeper object.
- Create an array (with four elements) to store object references of type Animal.
- Create the following objects and assign the object references to the next available array element.

Penguin p = new Penguin("Pengy",2,false);
Penguin p2 = new Penguin("Flapper",3,true);
Elephant e = new Elephant("Nelly",4, true);
Elephant e2 = new Elephant("Tiny",6, false);

- Iterate through the array, and determine whether each animal is hungry.
- If the animal is hungry, the Animal Keeper should feed it. Otherwise, the animal should perform its trick.

The expected program output is shown below.

```
Penguin Pengy Flap Wings
Penguin Flapper is being fed. He eats fish from a bucket!
Elephant Nelly is being fed, he eats hay..
Elephant Tiny blows with his trunk!
```

Please turn over

Q5 - JAT2Ex5

A music conductor conducts many instruments in an orchestra.

Write an application in Java to simulate the actions of a music conductor, conducting an orchestra.

Each instrument in the orchestra will play the “The Four Seasons’ by Vivaldi.

You must develop the solution as per the following design.

Playable Interface

Create an interface named **Playable**, which contains one abstract method:

- public String play(String piece);

The parameter of the method indicates the name of the piece to be played by an instrument.

Instrument class

Create a class to model an instrument. The Instrument class should implement the Playable interface and be marked as abstract (as the class will not implement the play() method).

The class should contain the following attributes:

- private String name;
- private String sectionName;
- private float weight;

The following constructors should be created:

- A constructor with three parameters (String, String, float).
- A constructor with zero parameters. If an object is created using this no-arg constructor, the following values should be passed to the 3-arg constructor (“No Instrument Name","No Section Name", 0).

Accessor methods (Setters and Getters) should be set for each of the instance variables.

An instrument cannot be assigned a negative weight. If an attempt is made to create an object with a negative weight, the weight attribute should be assigned a value of zero.

SubClasses

The orchestra will be made up of a number of instruments.

Create the following subclasses.

- Violin
- Clarinet
- FrenchHorn

Each class should contain the following:

- A constructor that accepts three arguments (String name, String sectionName, float weight).

- A constructor that accepts no arguments. This constructor should call the other (overloaded constructor) and pass it the following values ("Unknown Instrument Name","Unknown Section Name",0).

Please turn over

- A method named **play**(). The method should have one parameter (a variable of type String to indicate the name of the piece of music to play). The method should return a String, to simulate the action of the instrument playing the piece.

- **Example:**
  - public String play(String piece) {
    return("Violin playing " + piece + ".");
    }

MusicConductor Class

Create a class to model a MusicConductor.

The MusicConductor will conduct the orchestra in playing the specified piece of music.

Using polymorphism, come up with an optimal design for the MusicConductor class, which allows the Music Conductor to **conduct** any instrument (the ability to call the **play()** method of any instrument).

Main Class

Create a class named Main, which contains the main method.

The following objects should be created. The object references should be stored in an **arraylist**.

Attribute	Value
name	Violin
sectionName	Strings
weight	2.34f

Attribute	Value
name	Clarinet
sectionName	Woodwind
weight	1.45f

Attribute	Value
name	French Horn
sectionName	Brass
weight	5.43f

Create a MusicConductor object.

Request the Music Conductor to conduct each instrument in turn.

Create a folder named **JAT2Ex5** to store the requested interface and classes.

The expected output of the program is show below.

```
Violin playing The Four Seasons by Vivaldi.
Clarinet playing The Four Seasons by Vivaldi.
French Horn playing The Four Seasons by Vivaldi.
```

Please turn over

Q6 - JAT2Ex6

In this exercise, you will test your understanding of casting object reference variables.

Work through the logic of the following program and answer the questions that follow.

For each question, answer the following:

- Will the line of code result in a compiler error?
- Will the line of code result in a runtime error?

Use a pen and paper to record your answers. Are you correct? Create the program and find out!

```
public class Animal{}
```

```
public class Horse extends Animal{}
```

```
public class JAT2Ex6{

  public static void main(String[] args){
      Animal a = new Animal();

      Horse h = new Horse();

      Animal any = new Horse();

      Horse happy = a;

      Horse jack = (Horse) a;

      Horse hop = any;

      Horse hip = (Horse) any;

      Horse henny = (String) a;
  }// main

}// class
```

Q1: Animal a = new Animal(); • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q2: Horse h = new Horse(); • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q3: Animal any = new Horse(); • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.

Please turn over

Q4: Horse happy = a; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q5: Horse jack = (Horse) a; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q6: Horse hop = any; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q7: Horse hip = (Horse) any; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q8: Horse henny = (String) a; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.

Create a folder named **JAT2Ex6** to store the classes created.

Please turn over

Q7 - JAT2Ex7

In this exercise, you are once again tested on your understanding of casting object reference variables.

Work through the logic of the following program and answer the questions presented. Use a pen and paper initially to record your answers. Are your answers correct? Create the program and find out!

For each question, answer the following:

- Will the line of code result in a compiler error?
- Will the line of code result in a runtime error?

```
public interface Choppable{
      public String chop();
}
```

```
public class Tree implements Choppable{
  public String chop(){
    return "Chopping tree wood.";
  }
}
```

```
public class Oak extends Tree{
  public String chop(){
    return "Chopping oak wood.";
  }

  public String fell(){
    return "Felling an oak tree.";
  }

}
```

```
public class Bicycle{}
```

```
public class JAT2Ex7{
    public static void main(String[] args){
        Tree tree = new Tree();
        Oak o = new Tree();
        Tree t = new Choppable();
        Tree t2 = new Oak();

        Choppable c = new Oak();
        System.out.println(t2.chop());
        System.out.println(t2.fell());
        Oak t3 = (Oak) t2;

        System.out.println(t3.fell());
        Bicycle b1 = new Bicycle();
        Tree t4 = b1;
        Oak t5 = (Oak) tree;
        Oak t6 = (String) tree;
    }
}
```

Please turn over

Q1:
Tree tree = new Tree();
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q2:
Oak o = new Tree();
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q3:
Tree t = new Choppable();
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q4:
Tree t2 = new Oak();
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q5:
Choppable c = new Oak();
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q6:
System.out.println(t2.chop());
- True or false: Compiles and outputs, "Chopping oak wood."

Q7:
System.out.println(t2.fell());
- True or false: Compiles and outputs: “Felling an oak tree."

Q8:
Oak t3 = (Oak) t2;
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q9:
System.out.println(t3.fell());
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q10:
Bicycle b1 = new Bicycle();
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Q11:
Tree t4 = b1;
- True or False: The code fails to compile.
- True or False: The code compiles but produces a runtime error.

Please turn over

Q12: Oak t5 = (Oak) tree; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.
Q13: Oak t6 = (String) tree; • True or False: The code fails to compile. • True or False: The code compiles but produces a runtime error.

Create a folder named **JAT2Ex7** to store the classes created.

Please turn over

Q8 - JAT2Ex8

In your role as a Junior Java Programmer, you are working as part of a software development team, developing a new computer game entitled, 'James Bond – For Your Eyes Only'.

You are writing the code for the car chase scene in which James Bond transforms his Aston Martin (car) into a submarine and subsequently goes underwater.

Here is the scene.

Two villains (each driving a Ferrari) are **accelerating** over a bridge. They safely reach the other side.

James Bond is chasing the two villains.

Bond himself, is being pursued by the local sheriff (driving a squad car).

The squad car **accelerates**, the sheriff **activates** the siren and shouts into his walkie talkie, "Cut him off once he crosses the bridge..I've got you now, boy!"

The drawbridge is starting to rise. Bond **accelerates** and approaches the bridge at speed. Half-way across, he presses a button on the dashboard, '**transformToSubmarine'**. The Aston Martin drops to the water, transforms into a submarine, and submerges below.

The expected output of the program is shown below.

```
Ferrari accelerating...
Ferrari accelerating...
SquadCar accelerating...
NEE-eu NEE-eu
Cut him off once he crosses the bridge..I've got you now, boy!
Aston Martin accelerating...
Going underwater...transform to submarine complete.
```

Create the following classes:

Car

The class should be marked public and abstract. It should contain one abstract method.

```
public abstract String accelerate();
```

AstonMartin

The class should be marked public and extend the Car class. It should contain the following methods.

```
public String accelerate(){
  return("Aston Martin accelerating...");
}

public String transformToSubmarine(){
  return("Going underwater...transform to submarine complete.");
}
```

Ferrari

The class should be marked public and extend the Car class. It should contain the following method.

```
public String accelerate(){
    return("Ferrari accelerating...");
}
```

Please turn over

Squad Car

The class should be marked public and extend the Car class. It should contain the following methods.

```
public String accelerate(){
   return("SquadCar accelerating...");
}

public String activateSiren(){
   return("NEE-eu NEE-eu");
}

public String useWalkieTalkie(){
    return ("Cut him off once he crosses the bridge..I've got you now, boy!");
}
```

JAT2Ex8

This class should contain the main method. Create **an arraylist** to store objects of type Car (and its subclasses).

Create the following objects and add them to the arraylist.

- Ferrari (two objects)
- SquadCar
- AstonMartin

Create an **enhanced for loop** to iterate through the arraylist.

Reproduce the story line:

- For the Ferrari objects, simply call the accelerate() method.
- For the SquadCar object, call the accelerate(), activateSiren() and useWalkieTalkie() methods.
- For the AstonMartin object, call the accelerate() and transformToSubmarine() methods.

Create a folder named **JAT2Ex8** to store the classes created.

Please turn over

JAT2Ex9

Review the classes that follow. Determine the following:

Do all classes compile and if so, what is the output?

Test your answer by creating the program using your favourite text editor / IDE.

```
public class Parent{
  String name = "name not set";

  public Parent (String s){
   this.name=s;
  }

  public void setName(String s){
   this.name=s;
  }

  public String getName(){
   return name;
  }
}
```

```
public class Child extends Parent{}
```

```
public class JAT2Ex9{
  public static void main (String args[]){
    new Parent("John Smith");
    Child c = new Child();
    System.out.println(c.getName());
  }
}
```

Create a folder named **JAT2Ex9** to store the classes created.

Please turn over

JAT2Ex10

Review the classes that follow. Determine the following:

Do all classes compile and if so, what is the output?

Test your answer by creating the program using your favourite text editor / IDE

```
public class Super{
  private int x;

  public void setX(int x){
   this.x=x;
  }

  public int getX(){
   return x;
  }

}
```

```
public class Child extends Super{
  public Child(){
   this(0);
  }

  public Child(int x){
   super(x);
  }
}
```

```
public class JAT2Ex10{

 public static void main(String[] args){
   Child c = new Child();
   System.out.println(c.getX());
 }
}
```

Create a folder named **JAT2Ex10** to store the classes created.

Please turn over

JAT2Ex11

Carefully review the following class.

```
public class JAT2Ex11{
   static int a;
   final static int b;
   int c;
   final int d;

   public JAT2Ex11(){
     System.out.println("In Constructor");
     System.out.println("a's value is: " + a);
     System.out.println("b's value is: " + b);
     System.out.println("c's value is: " + c);
     System.out.println("d's value is: " + d);
   }

   public static void main(String[] args){
     new JAT2Ex11();
     System.out.println("**************************");
     new JAT2Ex11();
   }

   static{
     a=1;
     System.out.println("In static initialiser (1)");
    }

    {
     c=3;
     System.out.println("In instance init block (1)");
    }

   static{
    b=2;
    System.out.println("In static initialiser (2)");
   }

   {
    d=4;
    System.out.println("In instance init block (2)");
   }
}
```

In your opinion, is the output of the program as stated below?

Confirm if you are correct by creating the program using your favourite text editor / IDE.

Create a folder named **JAT2Ex11** to store the class created.

Please turn over

```
In static initialiser (1)
In static initialiser (2)
In instance init block (1)
In instance init block (2)
In Constructor
a's value is: 1
b's value is: 2
c's value is: 3
d's value is: 4
**************************
In static initialiser (1)
In static initialiser (2)
In instance init block (1)
In instance init block (2)
In Constructor
a's value is: 1
b's value is: 2
c's value is: 3
d's value is: 4
```

Please turn over

JAT2Ex12

Carefully review the following classes.

```
public class Parent{
  static byte a;
  private byte b;

  public Parent(){
   System.out.println("In Parent constructor.");
   a = 5;
   b = 2;
  }

  {
   b = 1;
   System.out.println("In instance init block - Parent");
  }

  static{
   a = 2;
   System.out.println("In static initialiser block - Parent");
  }

  public byte getB(){
   return b;
  }
}
```

```
public class Child extends Parent{
  static byte c;
  private byte d;

  public Child(){
   System.out.println("In Child constructor.");
   c = 10;
   d = 20;
  }

  {
   c = 1;
   System.out.println("In instance init block - Child");
  }

  static{
   c = 17;
   System.out.println("In static initialiser block - Child");
  }

  public byte getD(){
   return d;
  }
}
```

Please turn over

```
public class JAT2Ex12{

  public static void main(String[] args){
    Child c = new Child();
    System.out.println(c.a);
    System.out.println(c.getB());
    System.out.println(c.c);
    System.out.println(c.getD());
  }
}
```

In your opinion, is the output of the program as stated below?

Confirm if you are correct by creating the program using your favourite text editor / IDE.

Create a folder named **JAT2Ex12** to store the class created.

```
In static initialiser block - Parent.
In instance init block - Parent.
In Parent constructor.
In static initialiser block - Child.
In instance init block - Child.
In Child constructor.
5
2
10
20
```

End of Exercises