1. The following is some code designed by J.Hacker for a video game. There is an Alien class to represent a monster and an AlienPack class that represents a band of aliens and how much damage they can inflict:

```
class Alien {
   public static final int SNAKE_ALIEN = 0;
   public static final int ORGE_ALIEN = 1;
   public static final int MARSHMALLOW_MAN_ALIEN = 2;

   public int type; // Stores one of the three above types
   public int health; // 0 = dead, 100 = full strength
   public String name;

   public Alien(int type, int health, String name)
   {
            this.type = type;
            this.health = health;
            this.name = name;
   }
}

public class AlienPack
{
   private Alien[] aliens;

   public AlienPack(int numAliens)
   {
            aliens = new Alien[numAliens];
   }

   public void addAlien(Alien newAlien, int index)
   {
            aliens[index] = newAlien;
   }

   public Alien[] getAliens()
   {
            return aliens;
   }

    public int calculateDamage()
   {
            int damage = 0;
            for (int i=0; i < aliens.length; i++)
            {
                     if (aliens[i].type == Alien.SNAKE_ALIEN)
                     {
                              damage += 10; // Snake does 10 damage
```

```
                }
                else if (aliens[i].type == Alien.OGRE_ALIEN)
                {
                    damage += 6; // Orge does 6 damage
                }
                else if (aliens[i].type == Alien.MARSHMALLOW_MAN_ALIEN)
                {
                    damage += 1; // Maarshmallow Man does 1 damage
                }
            }
            return damage;
        }
    }
```

The code is not very object-oriented and does not support information hiding in the Alien class. Rewrite the code so that inheritance is used to represent the different types of aliens instead of the "type" parameter. This should result in deletion of the "type" parameter. Also rewrite the Alien class to hide the instance variables by creating a method getAlienType() and getAlienTypeName() to display the alien type and creat a "getDamage" method for each derived class that returns the amount of damage the alien inflicts. Finally, rewrite the calculateDamage methoed to use getDamage and write a main method that tests the code.

2. In previous exercise, the Alien class was rewritten to use inheritance. The rewritten Alien class should be made abstract since there will never be a need to create an instance of it, only its derived classes. Change this to an abstract class and also make the getDamage method and calculateDamage method and getAlienType() an abstract method. Test the class from your main method to ensure that it still operates as expected.

3. Create a class named Movie that can be used with your video rental business. The Movie class should tract the Motion Picture Association of America (MPAA) rating (e.g., Rated G, PG-13, R), ID number, and movie title with appropriate accessor and mutator methods. And create an equals() method that overrides Object's equals() method, where two movies are equal if their ID number is identical. Next, create three additional classes named Action, Comedy and Drama that are derived from Movie. Finally, create an overridden method named calcLateFees that takes as input the number of days a movie is late and returns the late fee for that movie. The default late fee is RM 10/day. Action movies have a late fee of RM 8/day, comedies are RM 6/day and dramas are RM 4/day. Test your classes from a main method.

4. Extend the previous problem with a Rental class. This class should store a Movie that is rented, an integer representing the ID of the customer that rented the movie and an integer indicating how many days late the movie is. Add a method that calculates the late fees for the rental. In your main method, create an array of type Rental filled with sample data of all types of movies. Then, create a method named lateFeesOwed that iterates through the array and returns the total amount of late fees that are outstanding.