Questions and Exercises

These questions and exercises is an opportunity to see what you've learnt from the lecture as well as practice the new things we've been talking about. In other words, these questions and exercises are completely optional but it's recomended to do them. In the end of the document you will find the answers to the questions as well as possible solutions to the exercises, note that one can solve an exercise in different ways. There will also be some suggestions about what one could code if one want to continue with some more advanced things. These suggestions will not come with a possible solution and might include things that haven't been covered in the lecture.

Question 1
What restriction does an abstract class have?

Question 2
What is required when making a subclass to an abstract class?

Question 3
The class ArrayList allows you to make a list with a dynamic size, compared to a normal array where you have to define its length. The ArrayList is a generic class. What does this mean when you're creating a new instance(object) of ArrayList?

Exercise 1
Write an abstract class called GeometricFigure which contains two methods for its name and its color(just the name of the color). There should also be a method that prints out information about it(name, color and area). Make an abstract method that returns the area of the figure and then make the subclasses: rectangle, circle and triangle.

Exercise 2
Write a generic class called Triplet that has three generic types and has three parameters in the constructor(one for each type). Write a static method that checks if two Triplets are equal to each other(by checking if the three parameter values are the same).

Further explorations
Expand the inventory system in the game example(the files can found on the lecture page). The inventory should have different tabs for different things and the player should only be able to carry a specific weight, therefore add a weight to each item. Add the ability to drop items on the ground and also make items randomly spawn in the world.

Answers and solutions

Answer to Question 1
An abstract class can't be instantiated. In other words, you can't make an object of the abstract class. You can however make subclasses to the abstract class, which is what they are used for.

Answer to Question 2
When making a subclass to an abstract class you'll have to either make the subclass abstract as well or make sure that every single abstract method is being overridden. This is so every non-abstract class has all its methods completely defined, and therefore allows us to create an object of that class.

Answer to Question 3
Since the ArrayList is generic we'll have to give it a type when it's being created, in the case of the ArrayList it's what kind of items we want to store in it. To create an ArrayList that stores Strings one can do it like the following

```
ArrayList<String> myStrings = new ArrayList<String>();
```

Possible solution to Exercise 1

```
public abstract class GeometricFigure{

   private String name;
   private String color;

   public GeometricFigure(String name, String color) {
      this.name = name;
      this.color = color;
   }

   public String getName() {
      return name;
   }

   public String getColor() {
      return color;
   }

   public abstract int getArea();

   public final void printInformaton() {
         System.out.println(getName() + " with the color " + getColor() + " has the area " + getArea());
   }
```

```
}

public class Circle extends GeometricFigure {

   private int radius;
   public Circle(String name, String color, int radius) {
      super(name, color);

      this.radius = radius;
   }

   @Override
   public int getArea() {
      return (int)(Math.PI * radius * radius);
   }

}

public class Rectangle extends GeometricFigure {

   private int width;
   private int height;
   public Rectangle(String name, String color, int width, int height) {
      super(name, color);

      this.width = width;
      this.height = height;
   }

   @Override
   public int getArea() {
      return width * height;
   }

}

public class Triangle extends GeometricFigure {

   private int width;
   private int height;
   public Triangle(String name, String color, int width, int height) {
      super(name, color);

      this.width = width;
      this.height = height;
   }

   @Override
```

```
  public int getArea() {
    return width * height / 2;
  }

}

public class GeometricExample {

  public static void main(String[] args) {
    GeometricFigure[] figures = {
      new Circle("A hole", "Black", 10),
      new Circle("Something big", "Red", 200),
      new Triangle("Pizza slice", "of Cheese", 20, 40),
      new Rectangle("Piece of paper", "White", 57, 40)
    };

    for (GeometricFigure figure : figures) {
      figure.printInformaton();
    }
  }

}
```

Possible solution to Exercise2

```
public class Triplet<T, T2, T3> {

  private T value;
  private T2 value2;
  private T3 value3;

  public Triplet(T value, T2 value2, T3 value3) {
    this.value = value;
    this.value2 = value2;
    this.value3 = value3;
  }

  //Overriding the equals method in Object
  @Override
  public boolean equals(Object object) {
    if (object instanceof Triplet) {
      Triplet triplet = (Triplet)object;

      return
        triplet.value.equals(this.value) &&
        triplet.value2.equals(this.value2) &&
        triplet.value3.equals(this.value3);
    }else{
      return false;
```

```
        }
    }


}
```