

Practical 02

1.

```
public class Item {  
    protected int location;  
    protected String description;  
  
    public Item(int location, String description) {  
        this.location = location;  
        this.description = description;  
    }  
    public int getLocation() {  
        return location;  
    }  
  
    public void setLocation(int location) {  
        this.location = location;  
    }  
  
    public String getDescription() {  
        return description;  
    }  
  
    public void setDescription(String description) {  
        this.description = description;  
    }  
}
```

2.

```
public class Item {  
    protected int location;  
    protected String description;  
  
    public Item(int location, String description) {  
        this.location = location;  
        this.description = description;  
    }  
  
    public int getLocation() {  
        return location;  
    }  
}
```

```

        public void setLocation(int location) {
this.location = location;
        }

        public String getDescription() {
            return description;
        }

        public void setDescription(String description) {
            this.description = description;
        }
    }

//main

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

        Item item1 = new Item(123, "Some item description");

        System.out.println("Location: " + item1.getLocation());
        System.out.println("Description: " + item1.getDescription());
    }
}

```

3.

```

public class Main
{
    public static void main(String[] args) {
        Item itemsn = new Item(123, "Some item description");

        System.out.println("Location: " + itemsn.getLocation());
        System.out.println("Description: " + itemsn.getDescription());
    }
}

// public class Item {
protected int location;

```

```

protected String description;
// Constructor
public Item(int location, String description) {
    this.location = location;
    this.description = description;
}

// Getters and Setters (optional, but useful)
public int getLocation() {
    return location;
}

public void setLocation(int location) {
    this.location = location;
}

public String getDescription() {
return description;
}

public void setDescription(String description) {
    this.description = description;
}
}

```

4.

```

public class Main {
    public static void main(String[] args) {
        Item itemsn = new Item(123, "Some item description");
        System.out.println("Location: " + itemsn.getLocation());
        System.out.println("Description: " + itemsn.getDescription());

        item1.setLocation(456);
        item1.setDescription("Updated item description");

        System.out.println("Updated Location: " + itemsn.getLocation());
        System.out.println("Updated Description: " + itemsn.getDescription());
    }
}

```

```

public class Item {

```

```

protected int location;
protected String description;

public Item(int location, String description) {
    this.location = location;
    this.description = description;
}

public int getLocation() {
    return location;
}

public void setLocation(int location) {
    this.location = location;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}
}

```

5. public class Main {    public static void  
main(String[] args) {

```

    Monster monster01= new Monster(321, "Scary monster", 100);

    System.out.println("Location: " + monster01.getLocation());
    System.out.println("Description: " + monster01.getDescription());
    System.out.println("Health: " + monster01.getHealth());

    Monster monster02 = new Monster(123, "Creepy monster");

    // Accessing properties using getters of both Monster and Item classes
    System.out.println("Location: " + monster02.getLocation());
    System.out.println("Description: " + monster02.getDescription());
    System.out.println("Health: " + monster02.getHealth());
}
}

```

```

// public class Monster extends Item {
    private int health;
}

    public Monster(int location, String description, int health) {
super(location, description);
        this.health = health;
    }

    public int getHealth() {
        return health;
    }

    public void setHealth(int health) {
this.health = health;
    }
}

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

        Monster monster01 = new Monster(321, "Scary monster", 100);

        System.out.println("Location: " + monster01.getLocation());
        System.out.println("Description: " + monster01.getDescription());
        System.out.println("Health: " + monster01.getHealth());

        Monster monster02 = new Monster(123, "Creepy monster");
        System.out.println("Location: " + monster02.getLocation());
        System.out.println("Description: " + monster02.getDescription());
        System.out.println("Health: " + monster02.getHealth());
    }
}

```

6. public class Main {    public static void  
     main(String[] args) {

```

        Monster monster01 = new Monster(321, "Scary monster", 100);
        System.out.println("Location: " + monster01.getLocation());
        System.out.println("Description: " + monster01.getDescription());

```

```

        System.out.println("Health: " + monster01.getHealth());

        Monster monster02 = new Monster(123, "Creepy monster");

        System.out.println("Location: " + monster02.getLocation());
        System.out.println("Description: " + monster02.getDescription());
        System.out.println("Health: " + monster02.getHealth());
    }
}

```

```

public class Monster extends Item {
    private int health;

    public Monster(int location, String description, int health) {

        super(location, description);
        this.health = health;
    }

    public int getHealth() {
        return health;
    }

    public void setHealth(int health) {
        this.health = health;
    }
}

```

PART 02:

1. b) super
2. b) private
3. d) None of the Mentioned.
4. b) Packages
5. c) import pkg.\*
6. c) charAt()
7. c) length()

PART 03: Fill in the blanks using appropriate terms.

1. Real-world objects contain **state** and **behavior**.
2. A software object's state is stored in **fields**
3. A software object's behavior is exposed through **methods**.
4. Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as data **encapsulation**.
5. A blueprint for a software object is called a **class**.
6. Common behavior can be defined in a **superclass** and inherited into a **subclass** using the **extends** keyword.
7. A collection of methods with no implementation is called an **interface**.
8. A namespace that organizes classes and interfaces by functionality is called a **package**.
9. The term API stands for **Application Programming Interface**.