АПСТРАКЦИЈА

Апстракција је концепт ООП који се своди на занемаривање небитних детаља и уредсређивање на битне.

Основне карактеристике апстрактне класе:

- Апстрактна класа се дефинише преко кључне речи abstract
- **Апстракта класа не може бити инстанцирана**, тј. не може се креирати објекат типа апстрактне класе.
- Ако је барем једна метода дефинисана као апстрактна онда и класа мора бити апстрактна.
- Апстрактна класа не мора имати апстрактне методе.
- Апстрактна метода је празна метода, тј. метода која нема имплементацију, дакле, нема тело, нема ни празно тело.
- Класа која је изведена из апстрактне класе мора да реализује (имплементира) све апстрактне методе тј. да дефинише њихово тело.

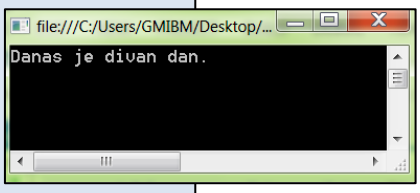
Пример како се како се дефинише апстрактна метода и како се имплементира у изведеној класи:

```
using System;

abstract class A {
    public abstract void prikazi();
}

class B : A {
    public override void prikazi()
    {
        Console.WriteLine("Danas je divan dan.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        B b = new B();
        b.prikazi();
        Console.ReadKey();
    }
}
```



ИНТЕРФЕЈСИ

Интерфејси су слични класи, али садржи само спецификацију а не и имплементацију својих чланова.

Основне карактеристике интерфејса:

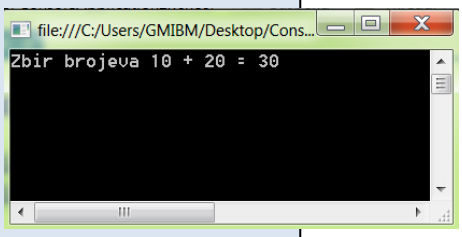
- За дефинисање интерфејса користи се кључна реч **interface**
- Интерфејс садржи само декларацију (спецификацију) методе али не и дефиницију (тело) методе
- Као апстрактна класа интерфејс се не може инстанцирати, самим тим не може садржати конструкторе и атрибуте.
- Сваки члан итерфејса подразумева се да је јаван (public) и апстрактан (abstract).
- Интерфејс омогућава вишеструко наслеђивање

Пример како се како се дефинише и имплементирају чланови интерфејса:

```
using System;

public interface Broj1 {
    int broj1();
}
public interface Broj2 {
    int broj2();
}
public interface Saberi : Broj1, Broj2 {
    void prikazi(int x, int y);
}
class implementacija: Saberi {
    public int broj1() { return 10; }
    public int broj2() { return 20; }
    public void prikazi(int x, int y) {
        Console.WriteLine("Zbir brojeva {0} + {1} = {2}",x,y,x+y);
    }
}

class Program
{
    static void Main(string[] args)
    {
        implementacija i = new implementacija();
        i.prikazi(i.broj1(), i.broj2());
        Console.ReadKey();
    }
}
```



АНАЛИЗИРАТИ ПРИМЕРЕ И ЗАОКРУЖИТИ ТАЧАН ОДГОВОР

234. Са леве стране дате су кључне речи које одређују типове класа, а са десне су описи класа. На линију испред описа уписати редни број под којим је наведен одговарајући тип класе:

1. abstract	_____	Класа која се простире у више фајлова
2. sealed	_____	Класа садржи само декларације метода, али не и дефиницију (тело) методе
3. partial	_____	Класа која се не може инстанцирати
4. interface	_____	Класа из које се не може наслеђивати

2

Када смо обрађивали наслеђивање, рекли смо да се кључна реч **sealed** користи када желимо да се одређене класе не наслеђује. Иако је за **abstract** и **interface** заједничко да не омогућавају инстанцирање, **interface** садржи само декларацију методе (нема тела методе) док то није случај са **abstract** класом.

Неке класе могу бити гломазне. У језику C# постоји могућност да се подели изворни код класе у неколико одвојених фајлова (датотека) тако да нека гломазна класа може да се организује као скуп неколико мањих класа. Дакле, када се нека класа подели у неколико фајлова, онда се дефинишу тзв. “парцијалне класе” помоћу кључне речу “**partial**”.

169.	Дат је код програма у програмском језику C# који дефинише интерфејс **Poredjenje**. Интерфејс Poredjenje садржи декларацију методе **porediPovrsine()**, која пореди објекте типа Figura. У датом коду дописати у 3.линији кода наредбу која недостаје да би метода била исправно декларисана. 1. namespace figure 2. 3. public_____{ 4. int porediPovrsine(Figura fig); 5. } Заокружити број испред одговора који даје исправно решење: 1. extends Poredjenje 2. interface Poredjenje 3. implements Poredjenje 4. abstract Poredjenje	**1**

extends – је кључна реч у Java - и, која се користи за наслеђивање. Дакле, у програмском језику C#, знак двотачка (:) користимо како би дефинисали наслеђивање док се у програмском језику Java користи кључна реч extends. Кључна реч **implements** је такође кључна реч из програмског језика Java и користи се када желимо да реализујемо неки интерфејс.

Очигледно је да одговор под 4. није тачан јер би недостаја кључна реч **class**.

228.	Метод дефинисан у родитељској класи, у класама наследницама може бити редефинисан или сакривен. Да би се омогућило **редефинисање** методе, при дефиницији у родитељској класи, испред ознаке повратног типа метода наводи се кључна реч **virtual**, **abstract** или **override**, док у изведеној класи испред ознаке повратног типа треба навести кључну реч _____. **Сакривање** методе родитељске класе врши се тако што се у изведеној класи испред ознаке повратног типа наведе кључна реч _____.	**1**

За редефинисање методе у изведеној класи поред ознаке повратног типа користи се кључна реч **override**. Док се сакривање методе родитељске класе, постиже додавањем кључне речи **new** испед ознаке повратног типа у изведеној класи.

209.	Да би наслеђени метод могао да се **редефинише** и тиме измени његова функционалност у класама наследницама, у родитељској класи испред ознаке повратног типа метода наводи се нека од понуђених кључних речи. Заокружити бројеве испред кључних речи које омогућавају редефинисање дефинисаног метода кроз ланац наслеђивања: 1. new 2. virtual 3. sealed 4. override 5. abstract 6. base 7. довољно је да буде public или protected	**1,5**

Ако погледате поставку 228. питања, можемо видети да нам се нуди одговор.

Са леве стране, у првој колони су наведени елементи, а у другој класе класификатори приступа елементима. Допуните реченицу која се односи на подразумевано право приступа наведеним елементима.

На линију у уписати одговарајуће појмове наведене у колонама:

1. Класа	1. private	Сви елементи наведени у првој колони
2. Поље класе	2. protected	са лева имају подразумевани
3. Својство у класи	3. public	класификатор приступа _____,
4. Метод класе	4. internal	осим _____ где је
5. Конструктор	5. protected internal	подразумевани приступ _____.

2

Подразумевани класификатор (модификатор) **приступа** за поља класе (атрибуте), методе класе, конструкторе и својства је **private**.

Подразумевани класификатор приступа за класу је: **internal**

На програмском језику C# дефинисане су две класе:

```
public class Racun {
   public virtual int Uvecaj() { return 10; }
}
public class Dinarski: Racun {
   public override int Uvecaj() { return 20 * base.Uvecaj(); }
}
public class Devizni : Racun {
   public override int Uvecaj() { return 50 + base.Uvecaj(); }
}
```

Унутар функције Main, креирана су три објекта ових класа на следећи начин:

```
Racun r = new Racun();
Racun rDin = new Dinarski();
Racun rDev = new Devizni();
```

Анализирати код и на предвиђене линије уписати шта метод Uvecaj() враћа при позиву из наведених објеката:

```
r.Uvecaj();      _____
rDin.Uvecaj();   _____
rDev.Uvecaj();   _____
```

3

Погледати 186. питање – логика је идентична