Из области алгебарских алгоритама, ученицима је могуће приказати и напредније алгоритме и њихове примене, као што су проширени Еуклидов алгоритам, одређивање модуларног инверза, алгоритам заснован на кинеској теореми о остацима. Још једна значајна група алгебарских алгоритама која се опционо може обрадити су алгоритми за рад са полиномима и великим бројевима (бројевима репрезентованим помоћу низа цифара). Свим ученицима је могуће приказати елементарну имплементацију основних операција над овим структурама података, а напреднијим ученицима је могуће приказати и ефикаснији Карацубин алгоритам множења заснован на техници подели-па-владај.

Из области геометријских алгоритама, могуће је приказати ученицима и напредније геометријске алгоритме, попут, на пример, одређивања конвексног омотача, ефикасног одређивања свих пресека у скупу дужи (line sweep алгоритам), одређивање пресека многоуглова и слично. Код напреднијих алгоритама детаљно анализирати сложеност.

Из области графовских алгоритама могуће је приказати и друге графовске алгоритме (нпр. Примов и Краскелов алгоритам, Дајкстрин алгоритам, Флојд-Варшалов алгоритам и слично).

Из области алгоритама за обраду текста ученицима је могуће приказати и ефикасне алгоритме претраге подниски (нпр. Кнут-Морис-Пратов, КМП алгоритам, Бојер-Муров алгоритам, Рабин-Карпов алгоритам и слично).

Израда пројектних задатака није обавезна у склопу овог предмета, али наставник може заинтересованим ученицима понудити теме пројектних задатака и оценити и ту њихову активност. Теме које се обрађују у склопу овог предмета се могу користити у склопу израде пројектних задатака из предмета „Објектно-оријентисано програмирање”.

III ПРАЋЕЊЕ И ВРЕДНОВАЊЕ НАСТАВЕ И УЧЕЊА

У процесу вредновања потребно је континуирано пратити рад ученика. У настави оријентисаној на достизање исхода вреднују се и процес и продукти учења. Прикупљање информација из различитих извора (свакодневна посматрања, активност на часу, учествовање у разговору и дискусији, самосталан рад, рад у групи, тестови) помаже наставнику да сагледа постигнућа (развој и напредовање) ученика и степен остварености исхода. Свака активност је добра прилика за процену напредовања и давање повратне информације. Важно је и ученике оспособљавати и охрабривати да процењују сопствени напредак у учењу.

У процесу праћења и вредновања значајну улогу имају домаћи задаци. Редовно задавање домаћих задатака (уз обавезну повремену проверу од стране наставника), помаже наставнику да стекне бољи увид у степен остварености исхода кроз анализу задатака које ученици нису умели да реше. Важно је и мотивисати ученике који редовно раде домаће задатке тако што ће њихов рад бити оцењен.

Вредновање активности у оквиру тимског рада се може обавити са групом тако да се од сваког члана тражи објашњење елемената урађеног рада и мишљење о сопственом раду унутар тима. Препоручује се да наставник са ученицима договори показатеље на основу којих сви могу да прате напредак у учењу, ученици се уче да размишљају о квалитету свог рада и о томе шта треба да предузму да би свој рад унапредили. Оцењивање тако постаје инструмент за напредовање у учењу. На основу резултата праћења и вредновања, заједно са ученицима треба планирати процес учења и бирати погодне стратегије учења.

Препоручено је да коначна оцена за сваког ученика буде добијена комбиновањем различитих начина оцењивања:

– активност на часу, учествовање у разговору и дискусији;
– редовна израда домаћих задатака;
– тестови - провера знања;
– пројектни рад, и појединачни и тимски.

Комбиновање различитих начина оцењивања помаже да се сагледају слабе и јаке стране сваког ученика. Приликом сваког вредновања постигнућа потребно је ученику дати повратну информацију која помаже да разуме грешке и побољша свој резултат и учење. Потребно је да наставник резултате вредновања постигнућа својих ученика континуирано анализира и користи тако да промени део своје наставне праксе.

ОБЈЕКТНО ОРИЈЕНТИСАНО ПРОГРАМИРАЊЕ

Циљ учења Објектно оријентисаног програмирања је стицање основних знања о објектно оријентисаној парадигми и њеној примени у решавању практичних проблема, развијање апстрактног и критичког мишљења и оспособљавање за примену стечених знања и вештина у даљем школовању и будућем раду.

ОПШТА ПРЕДМЕТНА КОМПЕТЕНЦИЈА

Учењем наставног предмета Објектно оријентисано програмирање ученик је оспособљен да примени стечена знања и вештине из области информационо-комуникационих технологија ради испуњавања постављених циљева и задатака у свакодневном животу, даљем школовању и будућем раду. Развио је способност апстрактног и критичног мишљења уз помоћ информационо-комуникационих технологија. Развио је дигиталну писменост и позитивне ставове према рачунарским наукама.

СПЕЦИФИЧНЕ ПРЕДМЕТНЕ КОМПЕТЕНЦИЈЕ

Специфичне предметне компетенције представљају опис специфичних способности ученика које му омогућавају да развије општу предметну компетенцију. Подразумевају способност за одговорно коришћење информационо-комуникационих технологија уз препознавање потенцијалних ризика и опасности. Специфичне компетенције обухватају способност да се разуме и примени начин решавања практичних проблема применом објектно оријентисане парадигме.

Разред	**Трећи**
Недељни фонд часова	**1 час теорије + 3 часа вежби**
Годишњи фонд часова	**37 часова теорије + 111 часова вежби**

ИСХОДИ По завршетку разреда ученик ће бити у стању да:	**ТЕМА** и кључни појмови садржаја програма
– наброји основне карактеристике објектно оријентисане парадигме; – употреби готове класе и објекте у креирању апликација; – наведе разлику између класе и објекта; – објасни поступак моделовања на конкретним примерима; – опише интерфејс задате класе; – демонстрира концепт енкапсулације и објасни права приступа елементима класе; – напише класу са потребним атрибутима и методама; – напише конструкторе и деструктор у класи; – осмисли и имплементира решење задатка коришћењем новодефинисане класе и њених објеката; – осмисли и имплементира класу коју затим користи у више различитих апликација; – за задати проблем креира једноставан систем повезаних класа и апликацију којом се тај проблем решава; – опише концепт наслеђивања и однос „врста-од"; – наброји примере неких наткласа и њихових изведених класа; – на примерима објасни права приступа елементима основне класе из објекта изведене класе; – дефинише конструкторе и деструкторе у наткласи и изведеним класама; – објасни принцип полиморфизма; – напише виртуалне методе у оквиру дефиниција класа; – дефинише апстрактне методе и апстрактне класе; – на примерима илуструје разлику између апстрактне класе и интерфејса; – осмисли и имплементира решење задатка коришћењем једне класе и класа изведених из ње; – за дати проблем уочи основне објекте и везе између њих, развије и имплементира хијерархије класа и интерфејса, помоћу којих могу да се реше тај и њему сродни проблеми; – тимски или индивидуално, а уз помоћ наставника, дефинише сложенији проблем за чије решавање осмишља и користи хијерархије класа; – тимски или индивидуално развије и приказује идејно решење проблема; – тимски или индивидуално развије план рада и начин праћења успешности реализације плана; – развије решење изабраног проблема или дела за који је задужен; – пише документацију; – креира презентацију и презентује решење пројектног рада; – вреднује своју улогу при изради пројектног задатка и активности за које је био задужен.	**ОСНОВНИ ПОЈМОВИ ОБЈЕКТНО ОРИЈЕНТИСАНОГ ПРОГРАМИРАЊА** Основне карактеристике објектно оријентисане парадигме. Проблеми који се решавају објектно оријентисаним приступом. Примена готових класа и објеката. Моделовање као основа за решавање проблема. Принцип апстракције у објектно оријентисаном програмирању (скраћено ООП). Класа и објекат. Инстанцирање класе. Улога и врсте конструктора, улога деструктора. Основни елементи класе: атрибути (поља) и методе Принцип енкапсулације у ООП, права приступа пољима и методама. Употреба креираних класа у више различитих апликација. Везе између класа. **ПРИНЦИПИ НАСЛЕЂИВАЊА И ПОЛИМОРФИЗМА** Наслеђивање. Наткласа и изведене класе (потклase). Поља и методе изведене класе, приступ компонентама основне класе Хијерархија класа. Улога и врсте полиморфизама. Виртуалне методе. Апстрактне методе и апстрактне класе. Интерфејси. Улога апстрактних класа и интерфејса. **ПРОЈЕКТНИ ЗАДАТАК** Фазе пројектног задатка од израде плана до представљања решења. Израда пројектног задатка . Презентовање идејног решења пројектног задатка. Презентовање и анализа решења пројектног задатка.

УПУТСТВО ЗА ДИДАКТИЧКО-МЕТОДИЧКО ОСТВАРИВАЊЕ ПРОГРАМА

Теоријски часови се изводе са целим одељењем. Препорука је да то буде у рачунарском кабинету и да ученик и током теоријских часова има активну улогу и може одмах да испроба једноставне примере. Настава вежби се изводи са половином одељења у рачунарском кабинету, у групама не већим од 12 ученика.

I. ПЛАНИРАЊЕ НАСТАВЕ И УЧЕЊА

Приликом планирања часа, исходе предвиђене програмом треба разложити на мање и на основу њих планирати активности за конкретан час. Треба имати у виду да се исходи у програму разликују, да се неки могу лакше и брже остварити, док је за одређене исходе потребно више времена, активности и рада на различитим садржајима. Исходе треба посматрати као циљеве којима се тежи током једне школске године.

При обради нових садржаја треба се ослањати на постојеће искуство и знање ученика, и настојати, где год је то могуће, да ученици самостално откривају математичке правилности и изводе закључке. Ученике треба упућивати да користе уџбеник и друге изворе знања, како би усвојена знања била трајнија и шира, а ученици оспособљени за примену у решавању разноврсних задатака.

На часовима треба комбиновати различите методе и облике рада, што доприноси већој рационализацији наставног процеса, подстиче интелектуалну активност ученика и наставу чини интересантнијом и ефикаснијом. Препоручује се коришћење интерактивних метода, пројектне, проблемске и истраживачке методе, дискусије, дебате и др., како би ученици били што више ангажованом током наставе. Комбиновати на часовима различите облике рада као што су самостални рад ученика (по принципу један ученик – један рачунар), рад у паровима (два ученика истовремено и заједно решавају конкретне задатке), рад у мањим групама (почетна анализа и идеје за методе решавања), као и рад са целом групом када наставник објашњава, приказује, демонстрира и кроз дискусију уводи ученике у нове области. Избор метода и облика рада, као и планирање активности ученика ускладити са наставним садржајем који треба реализовати на часу и предвиђеним исхода, али и са специфичностима одељења и индивидуалних карактеристикама ученика. Предложени број часова по темама је оквирни, на наставнику је да процени потребан и довољан број часова по темама узимајући у обзир знања и вештине који ученици имају из претходног школовања и животног искуства. Предложени редослед тема није обавезујући за наставнике, већ само представља један од могућих модела, који наставник може прилагодити у складу са изабраним програмским језиком и методолошким опредељењем.

Ради лакшег планирања наставе даје се оријентациони предлог броја часова по темама.

– Основни појмови објектно оријентисаног програмирања (50 часова)

– Принципи наслеђивања и полиморфизма (70 часова)

– Пројектни задатак (22 часа)

НАПОМЕНА: Предвиђена су два двочасовна писмена задатка са исправком (6 часова)

II ОСТВАРИВАЊЕ НАСТАВЕ И УЧЕЊА

Кроз обраду сваке теме ученици треба да што више буду активни и да током часова на рачунарима програмирају у конкретном изабраном објектно оријентисаном језику. Све теоријске појмове објаснити кроз конкретне примере класа и апликација у којима се користе објекти. Примери могу да буду једноставни, тако да се цела класа и апликација у којој се користе објекти кре-

иране класе може комплетно израдити на једном школском часу. Ставити акценат на апликације са графичким корисничким интерфејсом. Приказати бар неке примере са графиком (цртање, графички приказ објеката).

У оквиру теме **Основни појмови објектно оријентисаног програмирања** потребно је:

– Ученике укратко упознати са околностима и разлозима настанка објектно оријентисане парадигме.

– Анализирати основне карактеристике објектно оријентисане парадигме и објектно оријентисани приступ у решавању практичних проблема. Истаћи значај објектно оријентисаног програмирања (скраћено ООП) у изради већих пројеката на којима истовремено ради више програмера, као и значај ове парадигме у креирању софтверских компоненти (класа, или група повезаних класа) које се могу користити у различитим апликацијама (поновна употребљивост кода).

– Објаснити значај коришћења готових класа у савременом програмирању.

– Истаћи значај моделовања као основе за решавање проблема у оквиру објектно оријентисане парадигме. На конкретним примерима објаснити поступак моделовања – посматрање домена проблема, избор релевантних особина и добијање модела. Следе могући примери интерфејса задатих класа:

– Класа *Производ* са интерфејсом који обухвата очитавање цене (*Цена*), промену цене (*ПромениЦену*), проверу којој врсти производ припада (*ВрстаПроизвода*), проверу да ли је производ траженог произвођача (*Произвођач*), приказ података (*Приказ* или *ToString*) и слично. Ова класа може касније да се искористи као базна класа хијерархије различитих типова производа.

– *Аутомобил*, која треба да моделира кретање аутомобила. Корисник класе (возач) може да очита положај аутомобила, али не може произвољно да мења тај положај, тј. не може да премести аутомобил као играчку. Могуће команде, поред очитавања положаја, су: усмери се у датом смеру, повећај или смањи брзину за дату вредност, заустави се, крећи се током кратког времена (израчуна се нови положај) и слично. Кретање може да буде дуж праве линије, или по равни.

– Кроз одабране примере упознати ученике са основним принципима ООП: апстракција, енкапсулација, наслеђивање, полиморфизам. У даљем излагању ове теме посебно се осврнути и детаљно илустровати принципе апстракције и енкапсулације. Већ у процесу моделовања ученицима објаснити принцип апстракције, а енкапсулацију током креирања и примене класа. Посебна тема је посвећена принципима наслеђивање и полиморфизам, па те принципе у почетку изложити само укратко.

– Кроз одабране примере ученике упознати са основним појмовима објектно оријентисаног програмирања – класа и објекат.

– Објаснити основне елементе класе: атрибуте (поља) и методе, и њихову улогу.

– Објаснити однос између класе и објекта.

– Упознати ученике са готовим класама и објаснити њихов значај у изради објектно оријентисаних програма. Објаснити кроз примере појам, улогу и начин употребе готових генеричких класа из библиотеке.

– Упознати ученике са креирањем инстанци класе (објеката), животним веком објекта и преносом објеката као параметара метода:

– конструктори,

– деструктори.

– Анализирати начине и права приступа атибутима и методама. Обрадити са ученицима следеће теме:

– принцип енкапсулације (учауравања),

– јавни и приватни приступ елементима класе,

– дефинисање посебних метода за читање и постављање вредности атрибута тј. дефинисање својстава (ако их одабрани језик подржава),

– однос интерфејса класе и имплементације класе, значај њихове раздвојености, кроз примере илустровати промену имплементације без промене интерфејса

– Истаћи значај обраде изузетака. Објаснити механизам креирања и механизам обраде изузетка. Истаћи важност коришћења изузетака при креирању и модификовању објеката и у примерима користити изузетке кад год има смисла. На пример, објекат класе *Разломак* чији је именилац нула није исправан и у конструктору треба направити и испалити одговарајући изузетак и тиме спречити прављење неисправног објекта. Слично, у класи *Производ*, приликом модификовања цене направити и испалити одговарајући изузетак и тиме спречити постојање негативног броја као цене.

– Упознати ученике са заједничким (static) елементима класе, указати на њихове специфичности (како атрибута тако и метода). На пример, праћење броја инстанци класе, тј. броја креираних објеката, са циљем додељивања јединственог идентификатора сваком новом објекту. Илустровати концепт статичких класа (ако су подржане у одабраном програмском језику).

– Кроз једноставне примере упознати ученике са начином израде објектно оријентисаних програма. У почетку може да буде корисно да наставник понуди написану класу коју ученици треба да искористе у програму, или обрнуто, да наставник подели програм који се ослања на још ненаписану класу, а коју ученици треба да напишу. Ученици треба да буду што активнији у каснијим дискусијама кроз које се проблем моделира и смишља једна или неколико класа и начин њихове употребе. Како се напредује са реализацији различитих примера, тако ученици треба да постану што самосталнији у осмишљавању и имплементирању решења задатка коришћењем новодефинисане класе и њених објеката. Пожељно је да се понека класа употреби у више различитих апликација, да би се илустровала могућност поновне употребе написаног кода. Следе могући примери различитих класа и апликације које их користе.

– Класа *Особа*, са атрибутима име, презиме, година рођења, адреса и број мобилног телефона, и методама за упоређивање две особе по години рођења, по имену и презимену, за приказ особе, за промену адресе особе, промену броја телфона. Обратити пажњу да приликом креирања објекта година рођења особе не може да буде већа од текуће године, а касније не може да се мења, док се, на пример, контролисано могу изменити број телефона и адреса. Употреба може да се илуструје кроз апликације за издвајање података о особи из текстуалне датотеке, измену података о особи, претраживање особа, креирање одговарајућих спискова особа и слично.

– Класа *Производ* са атрибутима назив и цена, и методама за упоређивање са другим производом по цени (*СкупљиОд*, *ЈефтинијиОд*), промену цене (*ПромениЦену*) и приказ података (*Приказ* или *ToString*). Могуће је проширити класу са атрибутима назив произвођача, врста производа и слично и у складу са тим проширити и интерфејс. Апликација за приказ сортираног списка производа по цени. Апликација за претрагу списка производа (по називу, цени, произвођачу) и измену цена производа.

– Класа *Аутомобил* са апликацијама за анализу података о аутомобилима, продају аутомобила, претрагу аутомобила, и друге класе са сличним интерфејсом као описана класа *Производ*.

– Класа *Лоптица* са атрибутима положај (x и y координате), брзина кретања, величина и боја, и методама за цртање, покретање, промену брзине, промену смера кретања, заустављање, одбијање о други објекат или ивице. Апликације које имају једну или више лоптица које личе на једноставне рачунарске игрице или симулирају неки једноставан физички процес.

– Класа *Круг* која омогућава одређивање полупречника, површине, обима круга, проверу припадности тачке кругу, одређивање међусобног положаја два круга, померање круга, цртање круга и слично. Продискутовати шта су могући атрибути ове класе.

– Класе *Дуж*, *Квадрат*, *Правоугаоник*, *Троугао*, *Многоугао* и друге класе са сличним интерфејсом као описана класа *Круг*.

– Класа *КомплексанБрој*, апликације за манипулације са комплексним бројевима (могуће је са ђацима урадити и графичко представљање комплексног броја), на овом примеру истаћи различиту имплементацију класе без промене интерфејса (имплементације класе са реалним и поларним координатама).

– Класа *Време* (реализовати класу на више начина на пример са атрибутима сат и минут, и са атрибутом број минута од почетка дана) са основним методама за упоређивање два времена, одређивање времена после датог броја минута, приказ времена у различитим форматима (22:34, 10:34 PM) и слично.

– Класа *Датум* са основним методама, редни број дана у години, датум после к дана, датум пре к дана, упоређивање два датума и слично.

– Класа *Разломак* у којој су реализовати основне операције са разломцима, апликација за рад са разломцима (унос и избор операције, или рачунање вредности израза са разломцима).

– Класе којима реализујемо различите колекције целих бројева (на пример *Низ*/*Листа*, *Скуп*, *Стек*, *Ред*, …) при томе показати различите имплементације класа (на пример реализације стека коришћењем низа и коришћењем повезане листе).

– Уколико се у оквиру предмета *Програмирање* обрађује тема великих бројева, може да се имплементира класа *ВеликиПриродањБрој* у којој су реализоване основне операције за рад са природним бројевима произвољне дужине.

– Препорука је да се кроз примере ученици упознају са појмом и улогом генеричких класа. Са ученицима имплементирати примере генеричких класа (нпр. низ, стек, ред, скуп и слично).

– Упознати ученике са везама између класа тј. са класама чија су поља објекти других класа, или референцирају објекте других класа.

– Имплементирати са ученицима системе повезаних класа. Осмислити примере класа и апликација за интерактивну реализацију са ученицима на основу претходно урађених задатака. Кроз те примере ученици треба да се што више осамостале у решавању задатих проблема, креирањем једноставних система повезаних класа и апликација којима се проблеми решавају. Следе могући примери за интерактивну реализацију са ученицима.

– Коришћењем претходно дефинисаних класа *Време* и *Датум*, може да се имплементира класа *ВременскиТренутак* коју даље примењујемо у некој апликацији или другој класи.

– Имплементирати класе *Тачка*, *Вектор*, *Права* и користити их у решавању једноставних геометријских проблема (пожељно је обезбедити и цртање објеката).

– Класе *Моном* и *Полином*, са методама за рачунске операције над полиномима са више променљивих (класа *Моном* садржи низ слова која представљају имена променљивих и експонент уз свако име, а класа *Полином* садржи низ монома).

– Коришћењем претходно дефинисане класе *Особа* уз проширење по потреби, имплементирати класу *ВајберГрупа* (јединствени идентификациони број, име групе, администратор групе, списак особа – чланова…), креирати и класу *Порука* (особа и текст поруке) и обезбедити методе унутар класе *ВајберГрупа*, потребне за размену порука.

Тема **Принципи наслеђивања и полиморфизма** је централна тема предмета и за њу свакако треба одвојити укупно највећи број часова. У оквиру теме Принципи наслеђивања и полиморфизма потребно је:

– Упознати ученике са основним принципима наслеђивања (описати релацију ”је врста од”), начином креирања изведених класа, дефинисањем нових елемената у изведеној класи, креирањем конструктора за објекте изведених класа, правима приступа елементима основне класе у изведеној класи, као и начину редефинисања метода у изведеној класи.

– Објаснити принцип полиморфизма, виртуалне методе. Објаснити значење и разлике између статичког (у време превођења) и динамичког везивања (у време извршавања).

– Објаснити појам апстрактних метода и апстрактне класе.

– Објаснити појам интерфејса, декларацију и имплементацију интерфејса. Нагласити да је могуће да једна класа имплементира више интерфејса, као и да интерфејси могу да се наслеђују. Објаснити разлику између апстрактних класа и интерфејса.

– На конкретним примерима објаснити улогу апстрактних класа и интерфејса у хијерархији класа.

– Реализовати различите примере хијерархије класа у којима изведене класе поред понашања наслеђеног од базне класе имају и додатно, специфично понашање. Уз хијерархије класа реализовати и апликације које их користе. На пример:

– Класа *Особа* и изведене класе *Ученик, Професор, Директор, Помоћни Радник*. Све ове класе наслеђују основне атрибуте и методе од класе Особа и затим додају специфичне атрибуте и методе (на пример, просек оцена за ученика, одељење коме је разредни старешина за професоре и слично).

– Класа *Возило* и изведене класе *Путничко* и *Теретно*. Могуће је развити и класу *Гаража* као скуп возила (обезбедити улазак и излазак из гараже, као и евиденцију о слободним местима у гаражи у зависности од димензија возила). Слично, класа *Трајект* чува скуп возила и може да води рачуна о укупној маси (која се различито израчунава за путничка и теретна возила, јер се теретним возилима додаје маса терета, а путничким возилима маса путника).

– Класе потребне за пословање у банци (класа *Рачун*, различите врсте рачуна, класа *Трансакција*).

– Реализовати комплетне примере (динамичког) полиморфизма, тј. хијерархије класа у којој базна класа има један или више апстрактних метода, различито имплементираних у изведеним класама. На пример:

– Класа *Облик са* апстрактним методима *Обим*, *Површина*, *ПрипадностТачке*, *Транслација* и изведене класе *Троугао*, *Квадрат*, *Круг*.

– Класа *ТелефонскиПретплатник* који садржи податке о особи, број телефона, евиденцију о обављеним разговорима и објекат класе *ТарифниПакет* који на основу евиденције позива израчунава износ рачуна. *ТарифниПакет* има више изведених класа (на пример *Припејд* и *Постпејд*). Могуће је различито тарифирати разговоре у истој и различитој мрежи, домаћи и инострани саобраћај и слично.

– Класа *Израз* са апстрактним методом *ВредностУТачки* и изведене класе *Константа*, *Променљива*, *Збир*, *Разлика*, *Производ*, *Количник*. Хијерархију је могуће проширити и класом *Функција* и из ње изведеним класама *Логаритамска*, *Синусна, Косинусна*, итд. Класе којима је потребан аргумент (то су класе изведене из класе *Функција*) или два аргумента (класе операције: *Збир*, *Разлика*, *Производ*, *Количник*) садрже одговарајући број референци на класу *Израз*.

– Реализовати са ученицима неколико апликација, у којима се дефинише и користи неколико хијерархија класа које се комбинују у изради коначног решења. Пожељно је да се неке развијене хијерархије класа употребе у више различитих апликација, да би се илустровала могућност поновне употребе написаног кода. Могуће је приказати креирање пројекта у виду библиотеке (статичке или динамичке) чијим се укључивањем у решење избегава потреба за понављањем и поновним превођењем изворног кода у ком су дефинисане класе које се користе у више пројеката.

– Кроз веће задатке је пожељно илустровати основне принципе квалитетног објектно-оријентисаног дизајна: програмирање према интерфејсу, а не према имплементацији, учауравање и издвајање у засебне класе делова апликације који могу да варирају, давање предности композицији у односу на наслеђивање, креирање група класа (модула, библиотека) са што мањим интерфејсом и тиме мањим спрезањем са класама ван групе, креирање класа које су отворене за проширивање, али затворене за модификацију, креирање малих класа које треба да имају само једну одговорност, ... Кроз веће задатке и примере је пожељно илустровати и неке пројектне обрасце који се користе у објектно-оријентисаном софтверу (али без инсистирања на упознавању ученика са теоријом и класификацијом пројектних образаца). На пример, хијерархије израза и функција су типичан пример обрасца Composite, при чему је исти образац могуће илустровати и кроз примере класа датотека и директоријум, затим ставка менија и мени и слично.

Кроз израду сложеног пројекта у оквиру теме **Пројектни задатак** повезати стечено знање (нпр. израда апликације за вођење евиденције у школама) и на тај начин упознати ученике са могућностима објектно оријентисаног програмирања.

Пројектни задаци треба да представљају искуствено блиске проблеме за чије се решавање користи једна или више хијерархија класа. Прецизирати термин за приказ идејног решења пре него што тим приступи практичном раду. Прецизирати и термин за презентацију коначног решења. Континуирано пратити на часовима рад ученика. Упутити ученика на даља истраживања додатних тема како у програмском језику тако и у области алгоритама.

Препоручују се следећи кораци у оквиру израде пројекта:

– Што прецизнија спецификација задатка: опис функционалности, интерфејс према кориснику (шта корисник може да ради, шта се приказује) – за опис може да се користи поређење са познатим програмима;

– У спецификацију може да уђе и листа могућих проширења, која не морају да буду урађена, али је пожељно да су предвиђена (ако утичу на дизајн);

– Класе које ће да постоје у програму, за сваку класу размислити шта осталим класама треба од ње. На основу ових предвиђених захтева се постављају интерфејси класа;

– Имплементације планираних класа;

– Тестирање сваког дела функционалности током имплементације, отклањање грешака (пожељни су тест модули);

– Спајање свих делова у целину, тестирање апликације кроз сценарија употребе (систематично испробавање функционалности апликације).

Дати редослед корака треба схватити као начин рада у идеалном случају. Мање одступања од наведених корака обично значи и мање проблема, али нормално је да се нпр. интерфејс неке класе или преправи током имплементације других класа које је користе, или да се неки делови програма тестирају само кроз коришћење целе апликације (без посебног тест модула).

За пројектни рад понудити неколико могућих начина реализације, тако да ученици у договору са наставником бирају начин рада (наставник одобрава и пројекат и начин рада):

– Ученици који нису довољно сигурни да би могли самостално да ураде пројекат, могу цео пројекат да раде у пару;

– Сваки ђак ради свој пројекат, а на почетку у паровима или мањим групама дискутују све пројекте те групе, помажу једни другима око дизајна/плана (које класе ће имати и са којим функционалностима, како те класе сарађују итд.);

– Ученик самостално ради цео пројекат;

– За пројекат који је нешто већи по обиму или комплекснији по структури, ученици могу да се организују у парове или мање тимове, да у оквиру пара или тима договоре дизајн, поделе посао уз прецизирање интерфејса, затим свако независно имплементира и тестира одређене класе, а на крају повежу делове и тестирају рад целе апликације.

У сваком начину организовања ученика потребно је да наставник верификује поједине фазе израде пројекта (опис задатка, дизајн класа), односно да да сугестије или коментаре. Уколико ученици раде у тимовима посветити пажњу изазовима тимског рада, охрабрити изражавање ставова и упутити како се врши подела улога и решавају могући проблеми.

III ПРАЋЕЊЕ И ВРЕДНОВАЊЕ НАСТАВЕ И УЧЕЊА

У процесу вредновања потребно је континуирано пратити рад ученика. У настави оријентисаној на достизање исхода вреднују се и процес и продукти учења. Прикупљање информација из различитих извора (свакодневна посматрања, активност на часу, учествовање у разговору и дискусији, самосталан рад, рад у групи, тестови) помаже наставнику да сагледа постигнућа (развој и напредовање) ученика и степен остварености исхода. Свака активност је добра прилика за процену напредовања и давање повратне информације. Важно је и ученике оспособљавати и охрабривати да процењују сопствени напредак у учењу.

У процесу праћења и вредновања значајну улогу имају домаћи задаци. Редовно задавање домаћих задатака (уз обавезну повремену проверу од стране наставника), помаже наставнику да стекне бољи увид у степен остварености исхода кроз анализу задатака које ученици нису умели да реше. Важно је и мотивисати ученике који редовно раде домаће задатке тако што ће њихов рад бити оцењен.

Вредновање активности у оквиру тимског рада се може обавити са групом тако да се од сваког члана тражи објашњење елемената урађеног рада и мишљење о сопственом раду унутар тима. Препоручује се да наставник са ученицима договори показатеље на основу којих сви могу да прате напредак у учењу, ученици се уче да размишљају о квалитету свог рада и о томе шта треба да предузму да би свој рад унапредили. Оцењивање тако постаје инструмент за напредовање у учењу. На основу резултата праћења и вредновања, заједно са ученицима треба планирати процес учења и бирати погодне стратегије учења.

Препоручено је да коначна оцена за сваког ученика буде добијена комбиновањем различитих начина оцењивања:

– активност на часу, учествовање у разговору и дискусији;

– редовна израда домаћих задатака;

– тестови – провера знања;

– пројектни рад, и појединачни и тимски.

Комбиновање различитих начина оцењивања помаже да се сагледају слабе и јаке стране сваког ученика. Приликом сваког вредновања постигнућа потребно је ученику дати повратну информацију која помаже да разуме грешке и побољша свој резултат и учење. Потребно је да наставник резултате вредновања постигнућа својих ученика континуирано анализира и користи тако да промени део своје наставне праксе.

БАЗЕ ПОДАТАКА

Циљ учења База података је стицање основних знања о техникама пројектовања база података као одговора на пословну потребу за информационим системима. Усвајањем концепата из области база података, ученик развија способност да програмира и користи упите за добијање тражених информација из база, прављење извештаја и дистрибуцију података.

ОПШТА ПРЕДМЕТНА КОМПЕТЕНЦИЈА

Учењем наставног предмета Базе података ученик је оспособљен да примени стечена знања и вештине из области информационо-комуникационих технологија ради испуњавања постављених циљева и задатака у свакодневном животу, даљем школовању и будућем раду. Развио је способност апстрактног и критичног мишљења уз помоћ информационо-комуникационих технологија. Развио је дигиталну писменост и позитивне ставове према рачунарским наукама.

СПЕЦИФИЧНЕ ПРЕДМЕТНЕ КОМПЕТЕНЦИЈЕ

Специфичне предметне компетенције представљају опис специфичних способности ученика које му омогућавају да развије општу предметну компетенцију. Подразумевају способност за одговорно коришћење информационо-комуникационих технологија уз препознавање потенцијалних ризика и опасности. Специфичне компетенције обухватају способност ученика да упозна концепт база података, њихову организацију, коришћење упита за добијање тражених информација из база, прављење извештаја и дистрибуцију података. Оне подразумевају и овладавање вештином и техникама пројектовања база података као одговора на пословну потребу за информационим системима. Специфичне предметне компетенције обухватају способност ефикасног коришћења програмирања и рада са базама података за решавање различитих проблема у даљем образовању, професионалном раду и свакодневном животу.