Radila: Aida Zametica
Klasa: DtoRequestController
Metoda: WBTesting_RemoveItem (modificirana verzija metode RemoveItem u svrhu WB testiranja)

```
public async Task<IActionResult> WBTesting_RemoveItem(int id)
{
    string userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
    if (userId == null)
    {
        throw new ArgumentNullException(nameof(userId));
    }

    Cart cart = null;

    foreach (var cartEntry in _context.Cart)
    {
        if (cartEntry.CustomerID == userId && cartEntry.ProductID == id)
        {
            cart= cartEntry;
            break;
        }
    }

    if (cart != null)
    {
         if (cart.ProductQuantity != 1)
        {
            cart.ProductQuantity--;
        }
        else
        {
            _context.Remove(cart);
        }

        await _context.SaveChangesAsync();
    }

    return RedirectToAction("Cart", new
    {
        discountAmount = 0,
        discountType = 1,
        discountCode = ""
    });
}
```

1. **Obuhvat iskaza/linija (Statement/Line coverage)**

TestInitialize koji će se koristiti za testne metode

```
private Mock<IDiscountCodeVerifier> discountCodeVerifierMock;
private Mock<ApplicationDbContext> dbContextMock;
private DtoRequestsController controller;
private string userId = "testUserId";
private int productId = 1;

[TestInitialize]
0 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public void TestInitialize()
{
    discountCodeVerifierMock = new Mock<IDiscountCodeVerifier>();
    dbContextMock = new Mock<ApplicationDbContext>();
    controller = new DtoRequestsController(dbContextMock.Object, discountCodeVerifierMock.Object);
}

10 references | Please sign-in to New Relic CodeStream to see Code Level Metrics | 12/12 passing
private Mock<DbSet<T>> GetDbSetMock<T>(List<T> data) where T : class
{
    var dbSetMock = new Mock<DbSet<T>>();
    dbSetMock.As<IQueryable<T>>().Setup(m => m.Provider).Returns(data.AsQueryable().Provider);
    dbSetMock.As<IQueryable<T>>().Setup(m => m.Expression).Returns(data.AsQueryable().Expression);
    dbSetMock.As<IQueryable<T>>().Setup(m => m.ElementType).Returns(data.AsQueryable().ElementType);
    dbSetMock.As<IQueryable<T>>().Setup(m => m.GetEnumerator()).Returns(data.GetEnumerator());
    return dbSetMock;
}
```

Potrebno je napisati **tri** testa kako bi se obuhvatile sve linije programskog koda.

- **Test 1** obuhvata slučaj u kojem je user null i dolazi do bacanja izuzetka

```
// written by : Aida Zametica
[TestMethod]
[ExpectedException(typeof(ArgumentNullException))]
0 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public async Task WbTesting_RemoveItem_WhenUserIdIsNull_ThrowsArgumentNullException()
{
    var mockHttpContextAccessor = new Mock<IHttpContextAccessor>();

    mockHttpContextAccessor.Setup(a => a.HttpContext.User.FindFirst(ClaimTypes.NameIdentifier)).Returns((Claim)null);

    controller.ControllerContext = new ControllerContext
    {
        HttpContext = new DefaultHttpContext { User = new ClaimsPrincipal() }
    };

    await controller.WBTesting_RemoveItem(0);
}
```

Pokrivenost postignuta ovim testom

```
1 reference | Please sign-in to New Relic CodeStream to see Code Level Metrics | 1/1 passing
public async Task<IActionResult> WBTesting_RemoveItem(int id)
{
    string userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
    if (userId == null)
    {
        throw new ArgumentNullException(nameof(userId));
    }

    Cart cart = null;

    foreach (var cartEntry in _context.Cart)
    {
        if (cartEntry.CustomerID == userId && cartEntry.ProductID == id)
        {
            cart= cartEntry;
            break;
        }
    }

    if (cart != null)
    {
        if (cart.ProductQuantity != 1)
        {
            cart.ProductQuantity--;
        }
        else
        {
            _context.Remove(cart);
        }

        await _context.SaveChangesAsync();
    }

    return RedirectToAction("Cart", new
    {
        discountAmount = 0,
        discountType = 1,
        discountCode = ""
    });
}
```

- **Test 2** obuhvata slučaj u kojem user nije null, cart nije null i kvantitet proizvoda je različit od 1

```
// written by : Aida Zametica
[TestMethod]
0 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public async Task WBTesting_RemoveItemWhenProductQuantityMoreThenOneDecreasesQuantity()
{
    var cartList = new List<Cart>
    {
        new Cart { CustomerID = userId, ProductID = 1, ProductQuantity = 2 },
        new Cart { CustomerID = "otherUserId", ProductID = 2, ProductQuantity = 1 },
    };

    var cartDbSetMock = GetDbSetMock(cartList);

    dbContextMock.Setup(d => d.Cart).Returns(cartDbSetMock.Object);

    var userMock = new Mock<ClaimsPrincipal>();
    userMock.Setup(u => u.FindFirst(ClaimTypes.NameIdentifier)).Returns(new Claim(ClaimTypes.NameIdentifier, userId));

    controller.ControllerContext = new ControllerContext
    {
        HttpContext = new DefaultHttpContext { User = userMock.Object }
    };

    var result = await controller.WBTesting_RemoveItem(1);

    Assert.IsInstanceOfType(result, typeof(RedirectToActionResult));

    var redirectResult = (RedirectToActionResult)result;
    Assert.AreEqual("Cart", redirectResult.ActionName);
    Assert.AreEqual(0, redirectResult.RouteValues["discountAmount"]);
    Assert.AreEqual(1, redirectResult.RouteValues["discountType"]);
    Assert.AreEqual("", redirectResult.RouteValues["discountCode"]);

    var removedCartItem = cartList.SingleOrDefault(c => c.CustomerID == userId && c.ProductID == productId);
    Assert.AreEqual(1, removedCartItem?.ProductQuantity, "The item quantity should have been decreased in the cart.");

}
```

Pokrivenost postignuta ovim testom

```
2 references | Please sign-in to New Relic CodeStream to see Code Level Metrics | 2/2 passing
public async Task<IActionResult> WBTesting_RemoveItem(int id)
{
    string userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
    if (userId == null)
    {
        throw new ArgumentNullException(nameof(userId));
    }

    Cart cart = null;

    foreach (var cartEntry in _context.Cart)
    {
        if (cartEntry.CustomerID == userId && cartEntry.ProductID == id)
        {
            cart= cartEntry;
            break;
        }
    }

    if (cart != null)
    {
        if (cart.ProductQuantity != 1)
        {
            cart.ProductQuantity--;
        }
        else
        {
            _context.Remove(cart);
        }

        await _context.SaveChangesAsync();
    }

    return RedirectToAction("Cart", new
    {
        discountAmount = 0,
        discountType = 1,
        discountCode = ""
    });
}
```

- **Test 3** obuhvata slučaj u kojem user nije null, cart nije null i kvantitet proizvoda je jednak 1

```
// written by : Aida Zametica
[TestMethod]
0 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public async Task WBTesting_RemoveItem_WhenProductQuantityEqualOne_RemoveProduct()
{
    var cartList = new List<Cart>
    {
        new Cart { CustomerID = userId, ProductID = 1, ProductQuantity = 1 },
        new Cart { CustomerID = "otherUserId", ProductID = 2, ProductQuantity = 1 },
    };

    var cartDbSetMock = GetDbSetMock(cartList);

    dbContextMock.Setup(d => d.Cart).Returns(cartDbSetMock.Object);

    var userMock = new Mock<ClaimsPrincipal>();
    userMock.Setup(u => u.FindFirst(ClaimTypes.NameIdentifier)).Returns(new Claim(ClaimTypes.NameIdentifier, userId));

    controller.ControllerContext = new ControllerContext
    {
        HttpContext = new DefaultHttpContext { User = userMock.Object }
    };

    dbContextMock.
        Setup(m => m.Remove(It.IsAny<Cart>())).Callback<Cart>((entity) => cartList.Remove(entity));

    var result = await controller.WBTesting_RemoveItem(1);

    Assert.IsInstanceOfType(result, typeof(RedirectToActionResult));
    var redirectResult = (RedirectToActionResult)result;
    Assert.AreEqual("Cart", redirectResult.ActionName);
    Assert.AreEqual(0, redirectResult.RouteValues["discountAmount"]);
    Assert.AreEqual(1, redirectResult.RouteValues["discountType"]);
    Assert.AreEqual("", redirectResult.RouteValues["discountCode"]);

    var removedCartItem = cartList.SingleOrDefault(c => c.CustomerID == userId && c.ProductID == productId);
    Assert.IsNull(removedCartItem, "The item should have been removed");
}
```

Pokrivenost postignuta ovim testom

```
3 references | Please sign-in to New Relic CodeStream to see Code Level Metrics | 3/3 passing
104     public async Task<IActionResult> WBTesting_RemoveItem(int id)
105     {
106         string userId = User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
107         if (userId == null)
108         {
109             throw new ArgumentNullException(nameof(userId));
110         }
111
112         Cart cart = null;
113
114         foreach (var cartEntry in _context.Cart)
115         {
116             if (cartEntry.CustomerID == userId && cartEntry.ProductID == id)
117             {
118                 cart= cartEntry;
119                 break;
120             } }
121
122         if (cart != null)
123         {
124             if (cart.ProductQuantity != 1)
125             {
126                 cart.ProductQuantity--;
127             }
128             else
129             {
130                 _context.Remove(cart);
131             }
132
133             await _context.SaveChangesAsync();
134         }
135
136         return RedirectToAction("Cart", new
137         {
138             discountAmount = 0,
139             discountType = 1,
140             discountCode = ""
141         });
142     }
```

2. **Obuhvat grana/odluka (Branch/Decision Line coverage)**

Nekoliko tačaka grananja gdje se mogu zauzeti različite putanje:

- Odluka 1: izjava ‘if (userId == null)’
- Odluka 2: petlja ‘foreach (var cartEntry in _context.Cart)’
- Odluka 3: ugnježdena izjava ‘if (cart.ProductQuantity != 1)’
- Odluka 4: izjava ‘if (cart != null)’ ili izjava ‘if (cartEntry.CustomerID == userId && cartEntry.ProductID == id)’

Potrebno je napisati **četiri** testa kako bi se obuhvatile sve grane programskog koda.

Test 1 (userId == null) i **Test 2** (userId != null) iz prethodnog primjera pokrivaju Odluku 1.
Test 2/3 (cart != null) i novi test koji ćemo napisati **Test 4** (_context.Cart == null) pokrivaju Odluku 2.
Test 2 (cart.ProductQuantity != 1) i **Test 3** (cart.ProductQuantity == 1) iz prethodnog primjera pokrivaju Odluku 3.

- **Test 4** obuhvata slučaj u kojem user nije null i _context.Cart je null pa samim tim i cart je null

```
// added for white box testing purposes
// written by : Aida Zametica
[TestMethod]
0 references | Please sign-in to New Relic CodeStream to see Code Level Metrics
public async Task WBTesting_RemoveItem_WhenCartIsNull_RedirectToAction()
{
    var cartList = new List<Cart> {};

    var cartDbSetMock = GetDbSetMock(cartList);

    dbContextMock.Setup(d => d.Cart).Returns(cartDbSetMock.Object);

    var userMock = new Mock<ClaimsPrincipal>();

    userMock.Setup(u => u.FindFirst(ClaimTypes.NameIdentifier)).Returns(new Claim(ClaimTypes.NameIdentifier, userId));

    controller.ControllerContext = new ControllerContext
    {
        HttpContext = new DefaultHttpContext { User = userMock.Object }
    };

    dbContextMock.Setup(m => m.Remove(It.IsAny<Cart>())).Callback<Cart>((entity) => cartList.Remove(entity));

    var result = await controller.WBTesting_RemoveItem(1);

    Assert.IsInstanceOfType(result, typeof(RedirectToActionResult));

    var redirectResult = (RedirectToActionResult)result;

    Assert.AreEqual("Cart", redirectResult.ActionName);
    Assert.AreEqual(0, redirectResult.RouteValues["discountAmount"]);
    Assert.AreEqual(1, redirectResult.RouteValues["discountType"]);
    Assert.AreEqual("", redirectResult.RouteValues["discountCode"]);
}
```

3. **Obuhvat petlji (Loop coverage)**

Kako ne bismo pisali sve testove, fokusirat ćemo se na **cartList** varijablu koja je od značaja za broj prolaza kroz jedinu petlju programskog koda. UserId također mora biti različit od null kako bi se testovi uspješno izvršili.

Prilikom obuhvata petlji potrebno je proći kroz sljedeće strategije:

1. **Preskoči unutrašnjost (tijelo) petlje**

   ```
   var cartList = new List<Cart> {};
   ```

2. **Uradi jedan prolaz kroz petlju**

   ```
   var cartList = new List<Cart>
    {
       new Cart { CustomerID = "otherUserId", ProductID = 1, ProductQuantity = 2 },
    };
   ```

3. **Uradi dva prolaza kroz petlju**

   ```
   var cartList = new List<Cart>
    {
       new Cart { CustomerID = "otherUserId", ProductID = 1, ProductQuantity = 2 },
       new Cart { CustomerID = "otherUserId", ProductID = 2, ProductQuantity = 1 },
    };
   ```

4. **Uradi slučajan broj prolaza kroz petlju**

```
var cartList = new List<Cart>
 {
    new Cart { CustomerID = "otherUserId", ProductID = 1, ProductQuantity = 2 },
    new Cart { CustomerID = "otherUserId", ProductID = 2, ProductQuantity = 1 },
    new Cart { CustomerID = "otherUserId", ProductID = 1, ProductQuantity = 2 },
    new Cart { CustomerID = "otherUserId", ProductID = 1, ProductQuantity = 2 },

 };
```

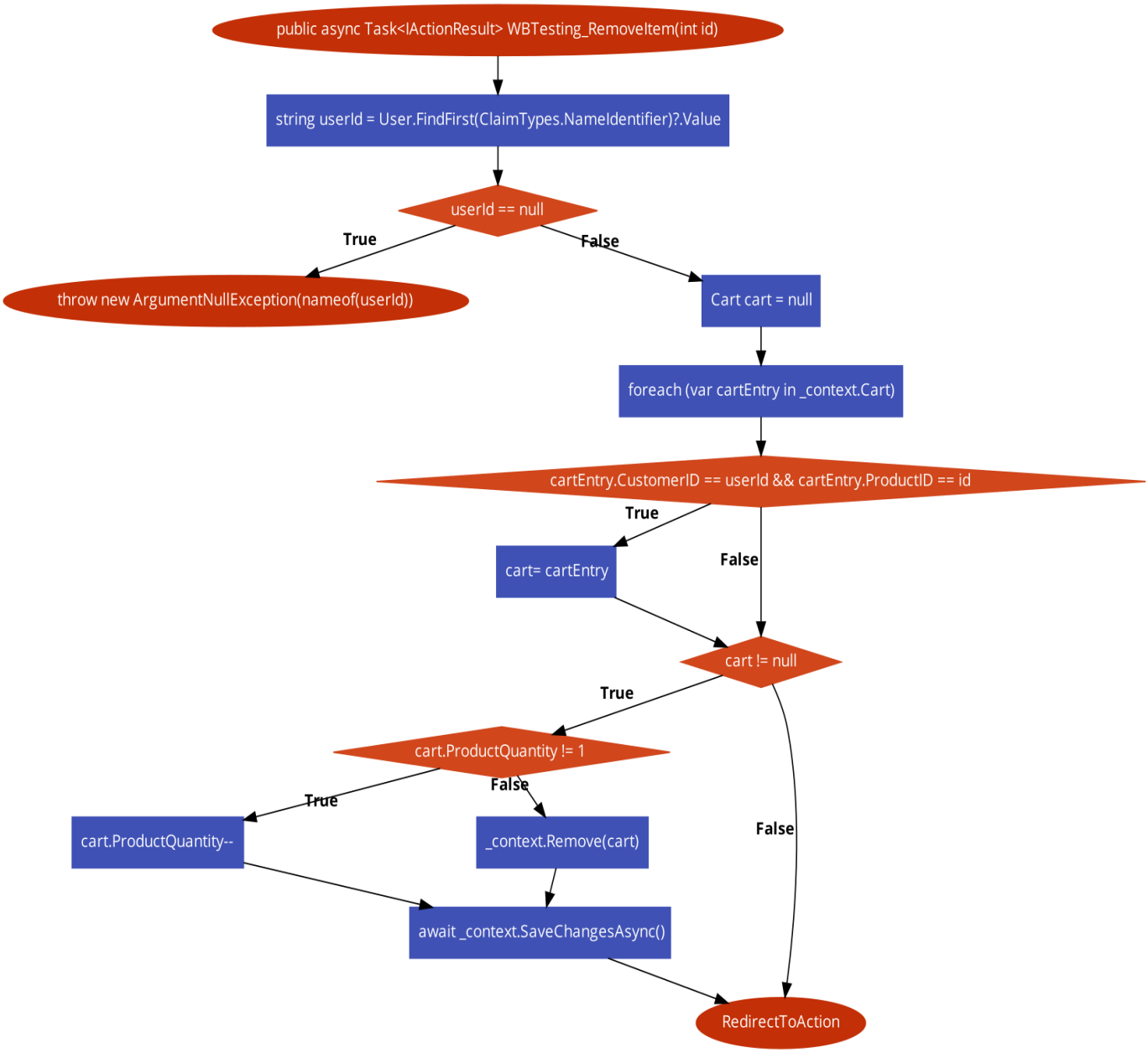
5. **Uradi n, n-1,n+1 prolaza kroz petlju n znači maksimalni broj prolaza kroz petlju**

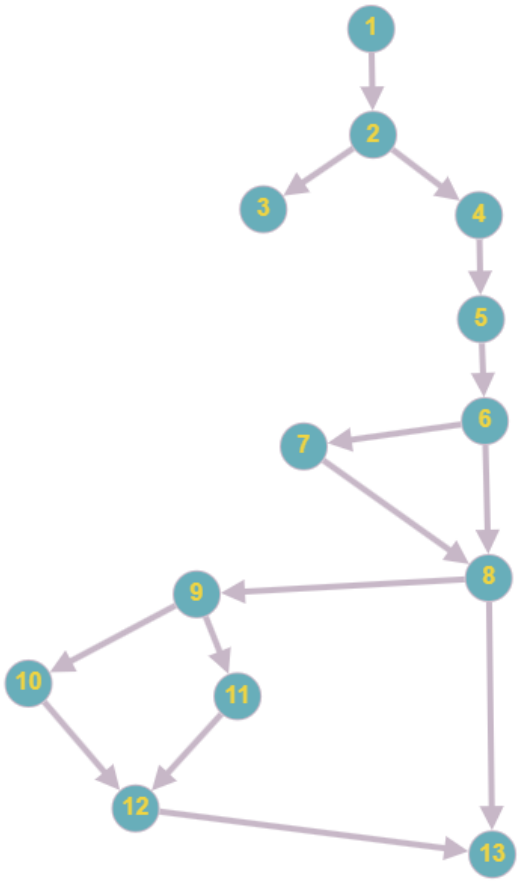
Nema smisla sa maksimalnim n probati za broj elemenata cartListe radi spriječavanja prekoračenja memorije ili drugih mogućih grešaka programa.

Potrebno je napisati **četiri testa** za potpuni obuhvat petlji izuzimajući posljednju strategiju.

Naglasit ću da je broj prolazaka kroz petlju moglo biti manipulisano linijom break koja se desi ispunjenjem uslova ‘cartEntry.CustomerID == userId && cartEntry.ProductID == id’

4. Obuhvat puteva (Path coverage)





Broj puta	Put
1	**1-2-3**

2	1-2-4-5-6-8-13
3	1-2-4-5-6-7-8-13
4	1-2-4-5-6-8-9-10-12-13
5	**1-2-4-5-6-7-8-9-10-12-13**
6	1-2-4-5-6-8-9-11-12-13
7	**1-2-4-5-6-7-8-9-11-12-13**

Uočava se 7 različitih puteva što pokazuje da je potrebno najmanje **sedam** testova da bi se postigla potpuna obuhvatnost puteva.

Boldirana tri testa su već napisana prethodno i pokrivaju proaktivno sve moguće pathove.

5. **Obuhvat uslova (Conditional coverage)**

Postoje 4 uslova unutar programskog koda koji mogu biti true ili false

- if (userId == null)
- if (cartEntry.CustomerID == userId && cartEntry.ProductID == id)
- If(cart != null)
- if (cart.ProductQuantity != 1**)**

Trebali bismo razmotriti sve moguće kombinacije istinitih i lažnih vrijednosti za ove uslove. Za četiri uslova bismo trebali imati 2^4 = 16 testova.

Ali pošto za userId == null ne izvršava se ostatak koda, možemo uzeti Test 1 koji je već napisan gdje je userId == null, a za ostale testove koristimo userId != null.

Zatim ako drugi uslov nije ispunjen neće biti ispunjen ni treći uslov pošto su uvezani.

Znači možemo uzeti :

- T, svejedno,svejedno,svejedno, svejedno (već napisan)
- F, F, F(zato što je i prethodni F), svejedno (pokrivaju ga naredna dva testa)
- F, T, T (zato što je i prethodni T), F (već napisan)
- F, T, T(zato što je i prethodni T), T (već napisan)